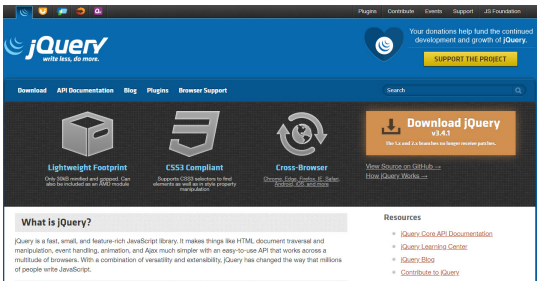
Javascript - jQuery

Importer



Avec jQuery, vous écrirez moins de code JavaScript, mais pour en faire toujours plus.

jQuery est le framework JavaScript le plus utilisé de tous. Complet, puissant, élégant... beaucoup sont ses points forts, et rares sont ses points faibles. L'attrait pour cet outil est immense.

Il y a 2 méthodes d'intégration de jQuery, le chargement du framework doit se faire avant tout autre script à la fin de la page avant la fermeture du ***</body>*** *:*

- *La première consiste à l'importer depuis un serveur externe, Google fourni un service de qualité, l'avantage est que l'importation se fait depuis un autre serveur qui potentiellement fournira d'autres site donc l'utilisateur a peut-être déjà ce fichier dans son cache. Rendez-vous à l'adresse :* ***https://developers.google.com/speed/libraries***

  <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

- *La seconde consiste à t'élécharger sur votre ordinateur depuis le site* ***jQuery.com*** *l'avantage est qu'il sera possible de travailler sur le projet sans être dépendant d'internet. La version* ***"compressed"*** *sera largement nécessaire, elle est plus légère et il ne sera pas utile de modifier le contenu du framework vu qu'il est possible d'ajouter des utilitaires depuis un autre fichier.*

  <script type="text/javascript" src="js/jquery-3.4.1.min.js"></script>

Le Sélecteur (jQuery ou $)

L'ensemble du framework jQuery repose sur une seule fonction. C'est la fonction la plus importante du framework, vous devrez toujours l'utiliser dès qu'il s'agira de développer en jQuery. Cette fonction, c'est tout bêtement la fonction ***jQuery()****, ou en plus simplifiée* ***$()*** *!*

Ainsi, pour agir avec les éléments d'une page web, on réalisera ce qu'on appelle couramment un « ciblage d'élément », qui représentera une balise, par exemple. On agira ensuite dessus grâce à des méthodes.

```
$("monElement").maMethode();
```

Pour cibler un élément, le plus simple est de mettre un ***sélecteur de type "CSS"***

```
$("p");                 // Sélectionne tous les paragraphes
$(".maClasse");         // Sélectionne les éléments ayant .maClasse pour classe
$("#monId");            // Sélectionne l'élément qui possède l'identifiant #monId
$("#monId .maClasse");  // Sélectionne les éléments ayant .maClasse pour classe dans l'élément qui possède l'ID #monId
```

Il est possible de ***rechercher à l'intérieur d'un élement*** *retourné avec la méthode "****FIND****", ce qui est très pratique lorsqu'un élément à été placé dans une variable*

```
var galerie = $("#galerie");        // Sélectionne l'élément avec l'ID "galerie" et le place dans une variable
galerie.find(".image");             // Sélectionne les éléments ayant .image pour classe dans l'objet "galerie"
```

$(this) = ?

Le sélecteur le plus spécial est sans aucun doute ***$(this)****. Dans la plupart des langages orientés objet, le mot-clé* ***this*** *représente l'objet courant, celui qui est actuellement traité par une fonction, par exemple. jQuery permet la sélection de cet objet pour y appliquer toutes les méthodes possibles avec jQuery.*

```
$("p").each(function(){             // La méthode each() permet de parcourir les élements retournés par le sélecteur
   $(this).html("Hello World !");   // $(this) représente le paragraphe courant
});
```

Les Événements

Pour expliquer la chose simplement, dites-vous qu'un événement est une action remplie par le navigateur, ou plus généralement l'utilisateur. ***On lance un bout de code quand une action est remplie****.*

L'événement le plus connu et ***ready()****, il s'execute lorsque le document est chargé, pour s'assurer que les scripts sont bien tous chargés avant d'effectuer la moindre modification et ainsi éviter de ne pas trouver l'objet ciblé.*

```
$(document).ready(function(){
   // Le code javascript ici !
});
```

L'utilisateur peut enclencher des événements par différentes actions tels que le clique, le double-clic, la mouvement de la souris, le survol, le scroll pour ne citer qu'eux.

```
$("#div1").click(function(e){ ... });          // Clique
$("#div2").dblclick(function(e){ ... });       // Double-clic
$("#div3").hover(function(e){ ... });          // Survol
$("#div4").scroll(function(e){ ... });         // Scroll
$("#div5").mousemove(function(e){ ... });      // Mouvement de la souris
```

L'utilisation du clavier peut aussi être récupéré avec les événements ***keyup****,* ***keypress*** *et* ***keydown*** *avec les entrées précises suivant une valeur unique pour chaque touche, caractère.*

```
$(document).keyup(function(touche){                // on écoute l'évènement keyup()
   var appui = touche.which || touche.keyCode;     // le code est compatible tous navigateurs grâce à ces deux propriétés
   if(appui == 13){                                // si le code de la touche est égal à 13 (ENTER)
      alert("Enter !");                            // on affiche une alerte
   }
});
```

Pour les ***formulaires*** *certaines actions sont à retenir :*

- ***focus()*** *lorsqu'on cible un champs*
- ***change()*** *lorsqu'on change la valeur d'un champs*
- ***submit()*** *lorsque le formulaire est envoyé*

De manière plus globale, la méthode ***on()*** *permet de les regrouper en une seule, elle est donc bien plus pratique et ergonomique !*

Un des avantages de cette technique est que l'écoute peut se faire sur plusieurs évènements en même temps, vous n'êtes pas obligé de créer un gestionnaire pour chacun d'eux ! Ainsi, nous pouvons lancer une écoute sur le clic et sur le double-clic, en séparant les deux types par un espace :

```
$("button").on("click dblclick", function(){          // on écoute le clic et le double-clic !
   alert("Ce code fonctionne !");
});
```

jQuery permet de ***simuler le déclenchement d'évènements*** *grâce à une simple méthode. Pour faire court, vous n'avez pas besoin d'attendre que l'utilisateur remplisse une action précise pour lancer du code : vous pouvez exécuter virtuellement un évènement grâce à* ***trigger()*** *! Il suffit de donner le type de l'évènement en tant qu'argument.*

```
$("#bloc").click(function(){
   alert("Cliqué !");
});
$("#bloc").trigger("click");          // déclenche l'action au chargement du script
```

Javascript - jQuery

Manipuler le code CSS

Parmi les nombreuses méthodes de jQuery, il en est une que vous utiliserez très souvent en ce qui concerne la modification du style de vos pages. Il s'agit de la méthode au nom très éloquent : css()

Cette méthode css() peut prendre plusieurs sortes d'arguments en parenthèses. Le plus simple est de spécifier le nom de la propriété, afin de récupérer sa valeur.

```
$("#bloc").css("color");    // Récupère la couleur dans la mesure où elle aura été définie pour cet objet spécifiquement
```

Il est aussi possible d'en définir et d'en modifier. Pour cela, il suffit de passer un deuxième argument à la méthode, qui contiendra la valeur à donner à l'attribut.

```
$("#bloc").css("color", "#ee3333");
```

Il est également possible de définir plusieurs propriétés CSS en même temps, grâce à un objet JavaScript que l'on passera en tant qu'argument. Il suffira de séparer les différents attributs par une virgule.

```
$("#bloc").css({
  color : "#ee3333",              // Change la couleur en "#ee3333"
  width : "300px",                // Assigne une largeur de 300px
  borderWidth : "1px",            // Ici, pas de désignation tel que le CSS classique, c'est un raccourci de "border-width"
  "border-left-width" : "10px"    // On peut utiliser la désignation CSS mais ne pas oublier de mettre entre parenthèses
});
```

Positionner des éléments

- ***offset()***, *qui définit la position d'un élément par rapport audocument, c'est donc une position absolue.*
- ***position()***, *qui définit la position d'un élément par rapport à son parent, c'est donc une position relative.*

Ces méthodes ne fonctionnent qu'avec deux objets, qui sont ***left*** *et* ***top****. Souvenez-vous que les données ont pour origine le coin en haut à gauche de la référence. Ainsi, pour récupérer la valeur de la position d'un élément :*

```
$("p").offset().left;      // retourne la valeur "left" de l'élément (position absolue)
$("p").position().top;     // retourne la valeur "top" de l'élément (position relative)
```

Il est possible de spécifier une nouvelle position à un élément, en passant par les méthodes précédentes. Il suffit de passer un objet en tant qu'argument, en donnant les nouvelles valeurs (en pixels) aux identifiantsleftettop:

```
$("p").offset({ left : 30, top : 200 });
$("p").position({ left : 200 });
```

Dimensions des éléments

- ***height()***, *qui retourne la hauteur formatée en pixels.*
- ***width()***, *qui fait la même chose avec la largeur.*
- ***innerHeight()*** *et* ***innerWidth()***, *qui prennent en compte les marges intérieures.*
- ***outerHeight()*** *et* ***outerWidth()***, *qui prennent en compte les marges intérieures et la bordure d'un élément.*

Le box model est un concept à connaître lorsque l'on manipule les dimensions avec jQuery. Conrètement, il faut retenir qu'il y a plusieurs types de largeur et d'hauteur. Les marges intérieures, les bordures et les marges extérieures sont des éléments à prendre en compte, et c'est pourquoi jQuery a créé plusieurs méthodes pour répondre à ce système.

```
$("p").height();              // retourne la hauteur stricte du paragraphe
$("p").innerWidth();          // retourne la largeur (avec marges intérieures) du paragraphe
$("p").outerWidth();          // retourne la largeur (avec marges intérieures + bordures) du paragraphe
$("p").outerHeight(true);     // retourne la hauteur (avec marges intérieures + bordures + marges extérieures) du paragraphe
```

Javascript - jQuery - Avancé

AJAX

*AJAX (Asynchronous JavaScript and XML) n'est pas une technologie, c'est le résultat d'un **ensemble de technologies du web** qui fonctionnent ensemble pour arriver à un objectif simple : rafraîchir une partie de page web sans recharger la page complète.*

Javascript a donc besoin de travailler en osmose avec un Serveur. On va appeler un fichier par une url relative ou absolue. On précise dans un objet les données à envoyer s'il y en a. On indique aussi si les données sont envoyées en POST ou en GET.

*Il faut encore lui dire comment interpréter les données récupérées, en **html**, **json**, **text**, etc... Pour au final préparer une fonction dans "success" et exécuter ce qu'on souhaite avec les données récupérées.*

```
$.ajax({
    url : "page.php",
    data : {
        id : 42
    },
    type : "GET",
    dataType : "html",
    success : function(code_html, statut){
        ...
    },
    error : function(resultat, statut, erreur){
        console.log(resultat);
    }
});
```

*Attention que la **fonction "success" ne s'exécutera qu'après le chargement des données**, le temps d'accès et de réponse du serveur créera une **latence** à prendre en compte dans le script.*

MAP

$.map() permet de gagner des lignes de code, il compile une procédure de boucle sur un tableau de données et la préparation de ces données dans un nouveau tableau.

```
var output = $.map(produits,function(produit){      // "produits" est un tableau, chaque ligne sera mise dans "produit"
  return {                                          // return permet d'insérer dans l'output une ligne supplémentaire
    id : produit.id,                                // la nouveau tableau contient un objet avec 2 références.
    title : produit.company.title+" - "+item.title
  };
});
```

En sortie, on obtient dans cet exemple une liste de produits avec juste un ID et un TITLE préparé spécialement poru l'occasion avec le nom de la société et le nom du produit.

Animate

*jQuery a une méthode permettant de modifier le CSS dynamiquement suivant une interpollation avec un temps donné pour créer une animation. L'avantage est qu'un événement se déclenche après l'animation. Par exemple pour faire disparaitre un objet avant de le supprimer complètement. Tous les paramètres sont accessibles à l'adresse : **https://api.jquery.com/animate/***

```
$( "#clickme" ).click(function() {                  // On lance l'animation lorsqu'on clique sur le bouton "#clickme".
    $( "#book" ).animate({
        opacity: 0,                                 // On modifie l'opacité et la hauteur.
        height: "200px"
    }, 5000, function() {
        $( "#clickme" ).show();                     // On réaffiche le bouton une fois l'animation terminée.
    });
    $( "#clickme" ).hide();                         // On cache le bouton pour éviter de cliquer plusieurs fois.
});
```