JQUERY

- Développement Web 420-427-RK
- Par Emeric Depierraz

Définition :

Jquery est une librairie Javascript qui “simplifie” la rédaction de scripts

Par défaut, les navigateurs Web connaissent JavaScript mais pas jQuery! Il s'agit donc d'une librairie JavaScript qu'il faut télécharger et inclure dans vos projets.

La librairie existe en 2 formats :

- La version **normale** (fichier facile à lire et à modifier au besoin, mais contenu volumineux)
- La version **minimisée** (fichier sur une seule ligne où tous les espaces et caractères inutiles ont été supprimés – difficile à lire)

Il y a 2 façons d'intégrer la bibliothèque à votre projet :

- À partir du fichier JavaScript de la librairie jQuery préalablement téléchargé dans votre projet.
- À partir d'un lien web permettant d'employer la bibliothèque (le fichier) à distance.

Javascript VS JQuery

JavaScript	jQuery
document.getElementByID("unID")	$('#unID') // sélecteur DOM
getElementByClassName("uneClasse")	$('.uneClasse') // sélecteur DOM
getElementByTagName("p")	$('p') // sélecteur DOM
myElement.textContent = "Allo";	myElement.text("Allo"); // modif. HTML
myElement.innerHTML = " Allo ";	myElement.html(" Allo "); // modif. HTML
var myText = myElement.textContent || myElement.innerText;	var myText = myElement.text(); // lecture HTML
var content = myElement.innerHTML;	var content = myElement.html(); // lire
myElement.style.display = "none";	myElement.hide(); // manip. CSS
myElement.style.display = "";	myElement.show(); // manip. CSS
myElement.style.fontSize = "35px";	myElement.css("font-size","35px");
element.parentNode.removeChild(element);	$("#id").remove(); // manip. DOM

Manipuler le HTML

Des méthodes jQuery pour manipuler le HTML :

- replaceWith() : permet de remplacer le contenu ET la balise HTML identifiée
- text() : permet de remplacer seulement le texte contenu dans une balise HTML identifiée
- append() : permet d'ajouter du contenu HTML juste avant la balise fermante identifiée
- prepend() : permet d'ajouter du contenu HTML juste après la balise ouvrante identifiée
- after() : permet d'ajouter du contenu HTML juste après la balise fermante identifiée
- before() : permet d'ajouter du contenu HTML juste avant la balise ouvrante identifiée

Manipuler le CSS ou le DOM

Des méthodes jQuery pour manipuler le CSS ou le DOM :

- remove() : permet de supprimer un ou des élément(s) du DOM
- attr() : permet de définir un attribut et sa valeur dans un élément du DOM ou de récupérer la valeur de l'attribut, si celui-ci existe :

```
var valeurAtt = $("#listItem").attr('style');

document.write(valeurAtt); // affiche "color:red"
```

- removeAttr() : supprime un attribut et sa valeur d'un élément HTML
- addClass() : ajoute une classe CSS à un ou des élément(s) du DOM
- removeClass() : supprimer une classe d'élément(s) du DOM
- toggleClass() : permet de donner ou de supprimer une valeur à une classe. Si la valeur existe, elle sera supprimée et si la valeur n'existe pas, elle sera ajoutée. Très pratique quand on veut appliquer des effets de style (ex: noir/blanc, ouvert/fermé, etc.)

Animer vos pages Web

Des méthodes jQuery permettent d'animer vos pages :

- show() : permet d'afficher un élément lentement, selon un nombre de millisecondes ou avec les paramètres "slow" et "fast". Cette méthode fonctionne avec le style "display"
- hide() : permet de cacher un élément lentement, selon un nombre de millisecondes ou avec les paramètres "slow" et "fast". Cette méthode fonctionne avec le style "display"
- fadeIn() : permet un effet de style et fait apparaître "en fondu" l'élément spécifié
- fadeOut() : permet un effet de style et fait disparaître "en fondu" l'élément spécifié

Animer vos pages Web (suite)

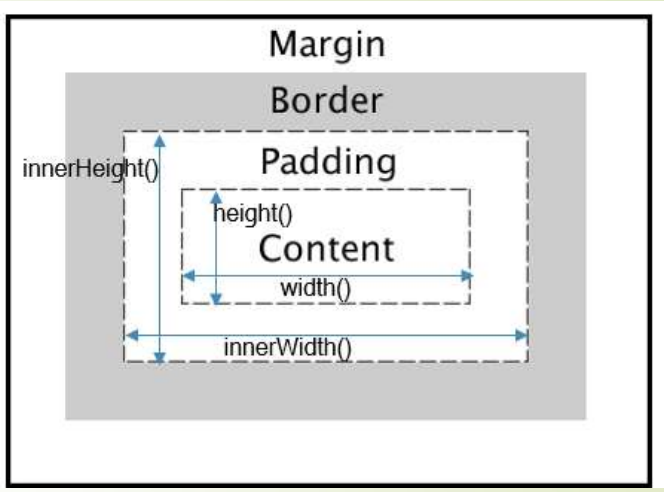
Des méthodes jQuery permettent d'animer vos pages :

- toggle() : permet de d'afficher ou de masquer le(s) élément(s) du DOM, selon son état actuel. On peut aussi lui passer en paramètre, la durée de l'effet
- fadeTo() : permet de contrôler le degré d'opacité de l'élément spécifié. La méthode utilise 2 paramètres: la durée et le degré d'opacité entre 0 et 1
- slideUp() : permet de déplacer vers le haut un élément spécifié
- slideDown() : permet de déplacer vers le bas un élément
- slideToggle() : permet de déplacer vers le haut ou vers le bas un élément spécifié (en alternance comme les autres "toggles")

https://www.w3schools.com/jquery/jquery_animate.asp

Tailles et positions

Il existe aussi des méthodes jQuery pour ajuster la taille des éléments :

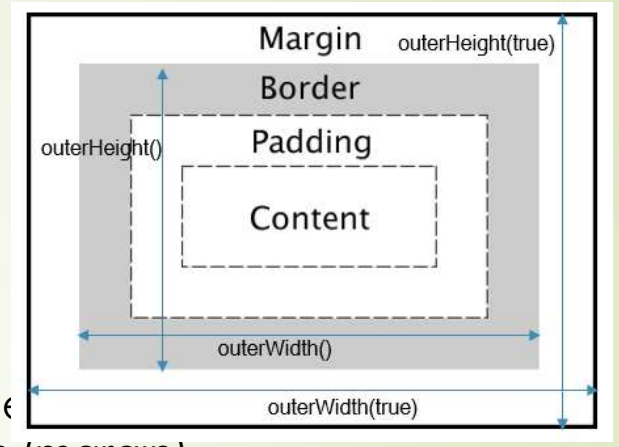


- height() et width() : permet de définir ou de récupérer la hauteu du DOM sans prendre en compte les marges, le padding ou la taille des bordures)
- innerHeight() et innerWidth() : permet de définir ou de récupérer la hauteur et la largeur d'un élément du DOM en prenant en compte le padding

https://api.jquery.com/height/

Tailles et positions (suite)

- outerHeight() et outerWidth() : permet de définir ou de récupérer la hauteur et la largeur d'un élément du DOM en prenant en compte la le padding et la taille des bordures
- outerHeight(true) et outerWidth(true) : permet de définir ou de récupérer la hauteur et la largeur d'un élément du DOM e padding, la taille des bordures et les marges (margin)
- offset() : permet de définir ou récupérer les coordonnées X et Y d'un élément DOM par rapport au coin supérieur gauche de la page
- position() : permet de définir ou récupérer les coordonnées X et Y d'un élément DOM par rapport au coin supérieur gauche de l'élément parent



Trouver des éléments du DOM

Plusieurs méthodes jQuery servent à parcourir le DOM. Voici un extrait de code HTML :

```
<div>
  <p>Premier paragraphe</p>
  <p><span>Deuxième</span> paragraphe</p>
  <p>Troisième paragraphe</p>
  <div id='d1'>Et un <span>dernier</span> paragraphe</div>
</div>
```

-
```
<script>
   $("p").find("span").css('color', 'red');
</script>
```

-
```
<script>
   $("span").parent().css('texteDecoration', 'underline');
</script>
```

-
```
<script>
     $("#d1").children().css('color', 'blue');
</script>
```

Résultat dans la page

Premier paragraphe
Deuxième paragraphe
Troisième paragraphe
Et un dernier paragraphe

Gérer les événements

```
<script>
   $("p").on("click", function()
        { traitement du clic});
</script>
```

Le méthode jQuery on("nomEvenement",function() {…}) facilite la gestion d'événements. Elle accepte comme premier argument :

- "click" : déclenché par un clic sur un élément du DOM (bouton gauche la de souris)
- "scroll" : déclenché lorsqu'on « déroule » la page ou un élément
- "hover" : déclenché quand le curseur de la souris survole un élément
- "mouseover" : déclenché quand le curseur de la souris passe sur un élément
- "mouseleave" : déclenché quand le curseur de la souris quitte un élément
- "keydown" : déclenché quand une touche est enfoncée et maintenue
- "keyup" : déclenché quand une touche est relâchée
- "keypress" : déclenché quand une touche est enfoncée et relâchée
- "focus" : déclenché quand un élément reçoit le focus
- "blur" : déclenché quand un élément perd le focus
- "resize" : déclenché quand la taille d'un élément est modifiée
- etc… : il existe bien d'autres événements. Au besoin, consulter la documentation officielle.

https://api.jquery.com/on/

Le paramètre event

Adaptons le HTML de la diapositive précédente :

```
<script>
   $("p").on("click", function(event)
        { traitement du clic};
</script>
```

L'événement comporte plusieurs méthodes :

- target : informe sur l'élément qui a déclenché l'évènement
- currentTarget : donne le type de l'élément déclencheur
- pageX et pageY : retourne les positions x et y de l'élément déclencheur
- timeStamp : retourne le temps depuis 1970/01/01, ce qui permet de déterminer la date et l'heure du déclenchement de l'évènement.

La fonction ready()

La fonction jQuery ready() est associée à un événement onload. Elle est déclenchée lorsque le navigateur a terminé de charger un composant dans le DOM.

```
<script>
        $(document).ready(function(){
            $("#content").load("content1.html");
        });
    </script>
```

Dans l'exemple précédent, la fonction anonyme associée à $(document).ready(); sera appelée lorsque le chargement de l'ensemble du document HTML sera terminé.

Ici, le code jQuery a été placé dans la balise <script> du document HTML. Toutefois, il aurait tout aussi bien pu se trouver dans un fichier JavaScript externe.

https://www.w3schools.com/jquery/event_ready.asp

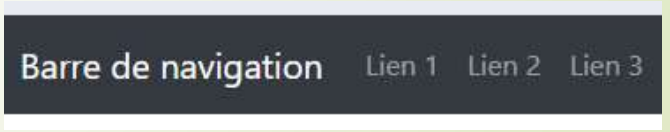
La fonction load()

La fonction jQuery load() est employée pour charger dynamiquement le contenu d'un fichier dans le document HTML courant.

```
<script>
        $(document).ready(function(){
            $("#content").load("content1.html");
        });
    </script>
```

L'exemple précédent montre que le contenu du fichier « content1.html » sera chargé uniquement lorsque le document HTML aura lui-même fini de se charger dans le navigateur.

En théorie, cette technique devrait permettre de charger un même menu dans plusieurs pages HTML.



https://www.w3schools.com/jquery/ajax_load.asp

Cross-origin resource sharing (CORS)

Depuis les années 2010, les navigateurs Web ont renforcé leur sécurité. La fonction load() de la page précédente ne peut donc pas être employée dans toutes les circonstances, au risque d'être bloquée.

Derrière ce blocage se cache CORS, littéralement le « partage de ressources entre origines multiples ». Concrètement, cela empêche que du code malveillant soit chargé dans votre navigateur à votre insu.

```
<script>
        $(document).ready(function(){
            $("#content").load("content1.html");
        });
    </script>
```

Dans l'exemple, votre navigateur suppose que « content1.html » pourrait justement contenir quelque chose de malveillant.

Conséquences du CORS

L'application du CORS sur votre ordinateur local vous empêche de mettre le menu d'un site Web dans un fichier que vous pourriez réutiliser dans plusieurs documents HTML.

Toutefois, si le même site Web était déployé avec un nom de domaine (par exemple monsiteweb.com) sur un serveur, le code pourrait fonctionner correctement !

Pour mieux comprendre, testez l'exemple suivant : https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax_load.

Ensuite, créez un fichier test.html sur votre ordinateur local et copiez à l'intérieur le code source provenant de l'exemple. En cliquant sur le bouton, vous constaterez que cette fois le code ne fonctionne pas dans votre navigateur !

Conclusions concernant CORS

- Votre navigateur Web est conçu pour protéger les internautes des attaques par injection de script, ce qui est une bonne nouvelle !
- En développement local, votre navigateur croit par défaut qu'un document chargé à partir d'un autre document n'appartient pas au même domaine et le bloque.
- Déployé sur un serveur avec un domaine comme www.monsiteweb.com, le même code n'est plus bloqué par votre navigateur car il respecte CORS.
- Pour parvenir à inclure un menu dans plusieurs pages d'un même site Web sur votre ordinateur local sans être bloqué par votre navigateur, nous devront employer un framework qui nous permet facilement de spécifier comment le navigateur doit appliquer la politique de sécurité CORS.