

Secteur Tertiaire Informatique
Filière « Etude et développement »

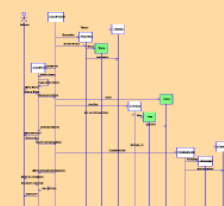
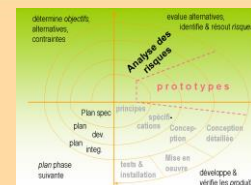
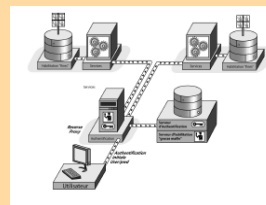
Séance « Développer des scripts clients dans une page web avec un framework »

UA : Ecrire un script JavaScript en utilisant les fonctions de base de JQuery

Apprentissage


Mise en situation

Evaluation



Version	Date	Auteur(s)	Action(s)
1.0	05/11/19	Lécu Régis	Création du document

TABLE DES MATIERES

Table des matières	2
1. Introduction	6
2. Premiers pas en jQuery	7
2.1 Télécharger jQuery	7
2.2 Utiliser la version en ligne	7
2.3 Quelques éléments de syntaxe	8
3. Gérer les événements HTML en jQuery	10
3.1 Une première fonction événementielle en jQuery	10
3.2 Liste des évènements	10
3.3 Séparer JavaScript et HTML	11
4. Gérer le DOM et les styles en jQuery	14
4.1 Manipuler le DOM en jQuery	14
4.2 Manipuler les styles CSS en jQuery	16
4.3 Parcourir l'arborescence du document avec jQuery	16
4.3.1 Rappels sur le DOM et les parcours d'arbre	16
4.3.2 Mise en pratique	17
4.3.3 Les méthodes de parcours d'arbre	17
5.  Exercice de synthèse	18
5.1 Cahier des charges	19
5.2 Première règle : séparer HTML et JavaScript	19
5.3 Deuxième règle : réaliser les traitements avec jQuery, étape par étape	20
5.3.1 Etape 1	20
5.3.2 Ajouter une nouvelle ligne sans présentation	21
5.3.3 Ajouter les règles de gestion sur les nouvelles lignes	21
5.3.4 Ajouter la gestion des modifications de prix et de quantité	22
5.3.5 Mettre à jour le total	24
6. Utiliser AJAX avec jQuery	25
6.1 Exemple 1 : avec un fichier local, par la méthode <i>load</i>	25
6.2 Exemple 2 : le même exemple avec une méthode de callback	26
6.3 Exemple 3 : un appel asynchrone avec une méthode de callback	26
6.4 Exemple 4 : récupérer du JSON	27

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

6.5	Pour aller plus loin avec AJAX en jQuery	28
-----	--	----

Objectifs

A l'issue de cette UA, le stagiaire sera capable de :

- Comprendre les avantages de jQuery par rapport à du code JavaScript sans Framework, en terme de facilité de codage et de compatibilité avec les navigateurs web
- Utiliser jQuery pour enrichir ses développements web

Pré requis

Pour suivre cet UA, le stagiaire doit posséder les bases d'HTML, de CSS et de JavaScript.

Pour le chapitre 6 qui traite d'AJAX en jQuery, le stagiaire doit posséder les bases d'AJAX.

Méthodologie

Ce document peut être utilisé en présentiel ou à distance.

Il précise la situation professionnelle visée par la séance, la situe dans la formation, et guide le stagiaire dans son apprentissage et ses recherches complémentaires.

Mode d'emploi

Symboles utilisés :



Renvoie à des supports de cours, des livres ou à la documentation en ligne constructeur.



Propose des exercices ou des mises en situation pratiques.



Point important qui mérite d'être souligné !

Ressources

- Site de référence de jQuery : <https://jquery.com/>
- Cours *w3schools* sur jQuery : <https://www.w3schools.com/jquery/default.asp>

1. INTRODUCTION

La devise de jQuery : « *write less, do more* »

jQuery est une bibliothèque JavaScript qui permet de simplifier énormément le codage du JavaScript dans les pages Web, en particulier les appels au **DOM** (*Document Object Model*) et l'utilisation d'AJAX.

De nombreuses tâches complexes en JavaScript vont se réduire à une seule ligne : l'appel d'une méthode jQuery, qui aura le même comportement sur l'ensemble des navigateurs Web.

jQuery fournit une « couche d'abstraction » au développement Web, qui évite au développeur de se préoccuper des différences entre les navigateurs, et simplifie ses tests.

Elle permet de :

- manipuler les éléments et attributs HTML, d'une façon plus simple et intuitive qu'avec le DOM
- manipuler les styles CSS
- gérer les événements HTML en leur associant des méthodes événementielles
- prendre en charge AJAX
- effectuer des effets graphiques et des animations (simples).

La bibliothèque jQuery fournit aussi des utilitaires et elle est évolutive (nombreux « *plugins* »)

2. PREMIERS PAS EN JQUERY

La bibliothèque jQuery est contenue dans un fichier **.js** que l'on peut au choix :

- Télécharger et insérer dans son application Web
- Utiliser en ligne, par un lien vers le site de jQuery

2.1 TELECHARGER JQUERY

<https://jquery.com/download/>

jQuery est fournie sous deux versions : une version compressée qui sera plus rapide en production, et une version décompressée, conseillée pour la mise au point pendant la phase de développement.

Téléchargez la version décompressée ([jquery-3.4.1.js](#)), et utilisez la dans une page de test, dans l'IDE de votre choix :

```
<!DOCTYPE html>
<html>
  <head>
    <title>test installation de jquery</title>
    <meta charset="UTF-8">
    <script src="jquery-3.4.1.js" type="text/javascript"></script>
  </head>
  <body>
    <script>
      alert ("jquery, version : " + $.fn.jquery );
    </script>
  </body>
</html>
```

Dans le script, **\$** désigne l'objet jQuery

\$.fn.jquery affiche la version de jQuery

2.2 UTILISER LA VERSION EN LIGNE

Remplacer le lien local par :

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

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

2.3 QUELQUES ELEMENTS DE SYNTAXE

jQuery a une syntaxe intuitive :

`$(selector).action ()`

- **\$** désigne la bibliothèque jQuery
- **selector** : critère de recherche pour « sélectionner » les éléments HTML sur lesquels va porter la méthode **action**
- **action** : ce qu'il faut faire sur le/les éléments sélectionnés

Un sélecteur peut être :

- l'élément HTML courant désigné par le mot clé *this* :

`$(this).action() ;`

- un élément HTML unique désigné par son id : comme dans un fichier CSS, l'id sera précédé de **#**

Par exemple, pour l'élément *idtest* :

`$("#idtest").action() ;`

- tous les éléments HTML d'un certain type : par exemple, toutes les *div*

`$("div").action() ;`

- tous les éléments HTML qui possèdent une certaine classe CSS : comme dans un fichier CSS, la classe sera précédée de **.**

Par exemple, pour la classe *titre*

`$(".titre").action() ;`

Quelques fonctions pour commencer :

hide / show : cache ou affiche un/plusieurs éléments HTML

text : retourne le texte des éléments sélectionnés (sans les balises HTML)

html : retourne le contenu des éléments sélectionnés (y compris les balises HTML)

val : retourne le contenu des champs texte, dans un formulaire

append : ajoute un élément fils à un container (comme *div*, *span*, *p*, *h1*, *table* etc.)

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »



Exercice 1 : Prise en main de jQuery

Dans cet exercice, les différentes opérations seront exécutées séquentiellement, et séparées par des « alert »

En partant de la page HTML **exo1.html** (dans **Livrables**, projet NetBeans **TestjQuery**)

- Cachez tous les paragraphes
- Affichez uniquement le paragraphe « p1 »
- Affichez le texte du paragraphe « p1 » puis changer son texte
- Affichez le contenu du paragraphe « p2 » (HTML compris)
- Ajoutez une image aux éléments qui ont la classe « titre »

Proposition de corrigés : **exo1.html** (dans **Corriges**, projet NetBeans **CorrigesjQuery**)



Exercice 2 : Essayez d'autres « sélecteurs » jQuery

Ce document AFPA est suffisant pour démarrer en jQuery.

Mais nous vous proposerons des compléments en nous appuyant sur le site *w3schools* qui propose un résumé de cours, et permet de visualiser, tester et modifier des exemples en ligne (« try it »)

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

More Examples of jQuery Selectors

Syntax	Description	Example
\$("#*")	Selects all elements	Try it
\$(this)	Selects the current HTML element	Try it
\$("#p.intro")	Selects all <p> elements with class="intro"	Try it
\$("#p:first")	Selects the first <p> element	Try it
\$("#ul li:first")	Selects the first element of the first 	Try it
\$("#ul li:first-child")	Selects the first element of every 	Try it
\$("#[href]")	Selects all elements with an href attribute	Try it
\$("#a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"	Try it
\$("#a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"	Try it
\$("#:button")	Selects all <button> elements and <input> elements of type="button"	Try it
\$("#tr:even")	Selects all even <tr> elements	Try it
\$("#tr:odd")	Selects all odd <tr> elements	Try it

- Exécutez les exemples proposés
- Ne vous préoccupez pas pour le moment de l'aspect événementiel (chapitre suivant).
- N'hésitez pas à modifier le code en ligne, pour faire vos propres essais.

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

3. GERER LES EVENEMENTS HTML EN JQUERY

Pour nos premiers pas en jQuery, nous avons utilisé un script séquentiel, en séparant les étapes par des « alert ».

Il est maintenant temps de revenir à la programmation événementielle, qui est très pratique en jQuery.

3.1 UNE PREMIERE FONCTION EVENEMENTIELLE EN JQUERY

Exemple : gestion de l'événement « click » sur tous les paragraphes, en lançant une fonction événementielle (anonyme) :

```
$("#p").click ( function()
{
    // traitement à effectuer
    alert ("cliqué") ;
    $(this).hide() ;
});
```

La fonction événementielle est passée en paramètre à la méthode *click*

Dans cet exemple, la fonction événementielle affiche « cliqué » puis cache le paragraphe qui a été cliqué, référencé par *\$(this)*



Exercice 3 :

Retirez le script séquentiel et complétez l'exercice 1, par cette première fonction événementielle

Proposition de corrigés : **exo3.html** (dans **Corriges**, projet NetBeans **CorrigesjQuery**)

3.2 LISTE DES EVENEMENTS

Ce sont les mêmes qu'en JavaScript :

Souris	Clavier	Formulaire	Document/Fenêtre
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload
mousedown			
mouseup			
mousemove			

Chaque événement est géré par une méthode jQuery du même nom : *click()*, *dblclick ()* etc.

La fonction événementielle reçoit le détail de l'événement par le paramètre *event* :

```
$("#").mouseenter( function (event)
{
    $("#trace").text ("=> entre dans : " + event.relatedTarget );
});
```

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Dans cet exemple, la méthode événementielle affiche l'élément HTML dans lequel on entre (*event.relatedTarget*) dans un élément "#trace"



Exercice 4 :

Sur le même principe, vous allez tester quelques événements avec leur paramètre *event* :

- Événements souris : *mousedown*, *mouseup*, *mousemove*

Afficher le type d'événement (*event.type*) et la position de la souris (*event.pageX*, *event.pageY*)

- Événements clavier : *keypress*, *keydown*, *keyup*

Afficher le type d'événement : *event.type*

Etc.

Vous pouvez vous appuyer sur le site *w3schools*, qui vous permettra d'essayer les événements avant de coder :

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

ou sur la documentation officielle de jQuery :

<https://api.jquery.com/category/events/>

Proposition de corrigés : **exo4.html** (dans **Corriges**, projet NetBeans **CorrigesjQuery**)

3.3 SEPARER JAVASCRIPT ET HTML

Notre code est-il propre ? Respecte-t-il les normes de codages usuelles ?

Deux points faibles :

- les appels à jQuery sont placés dans le fichier HTML : la préconisation actuelle est de séparer la description des contenus (HTML) et les traitements événementiels (fichier .js)
- notre exemple est dépendant de la place du script dans la page HTML : il fonctionne car il est placé après la définition des éléments HTML qui sont manipulés. Si on le place en début de fichier (dans le header), il ne fonctionne plus.

jQuery fournit un moyen pratique pour résoudre ces problèmes : l'événement *ready* se déclenche lorsque le document courant (ou un autre élément au choix) est complètement chargé. Comme les autres événements, il est géré par la méthode du même nom :

```
$(document).ready ( function()
{
    // traitements à effectuer après le chargement complet du document :
    // initialisation d'éléments HTML, création dynamique d'éléments par le DOM, association
    // des événements et des méthodes événementielles
});
```

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

L'exemple précédent devient :

```
$(document).ready ( function()
{
    $("p").click ( function()
    {
        // traitement à effectuer
        alert ("cliqué") ;
        $(this).hide() ;
    } ) ;
});
```

C'est la façon la plus répandue d'écrire le JavaScript.

Est-elle lisible pour autant ? Est-ce prudent dans un langage réputé pour ses failles de sécurité ?

Bonne pratique : utiliser des fonctions explicites pour les fonctions événementielles (comme les « *event listener* » en Java)

```
// méthode événementielle associée à l'événement click sur les
// paragraphes
function onParagraphClick()
{
    // traitement à effectuer
    alert ("cliqué") ;
    $(this).hide() ;
}
// cette fonction crée ou initialise tous les composants
// du document et leur associe des méthodes événementielles
function initComponents()
{
    // initialisations diverses
    // gérer les événements
    $("p").click (onParagraphClick) ;
}
// appelle la fonction d'initialisation après chargement du document
$(document).ready (initComponents) ;
```



Exercice 5 :

En repartant de l'exercice 4 :

- Vérifiez sur un exemple qu'un script placé dans le header ne peut pas modifier les éléments HTML qui sont déclarés après lui. Les sélecteurs génériques (*, class, balises) fonctionnent mais pas la recherche par id (#id)
- Placez les appels à jQuery dans un fichier .js, avec la démarche qui vient d'être expliquée :
 - commencez par l'exemple ci-dessus qui affiche « clique » puis efface le paragraphe, lorsque l'on clique dessus (première version avec les fonctions anonymes)
 - puis réécrivez le dans un style lisible, avec des méthodes intermédiaires *initComponents* et *onParagraphClick*

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

- Puis réécrivez l'exercice 4 en utilisant des méthodes intermédiaires, au lieu des fonctions anonymes :
 - le code doit rester lisible
 - affichez les *id* des objets concernés (*event.target.id* et *event.relatedTarget.id*)
 - différenciez les traitements selon les objets : par exemple, n'effacez que le paragraphe « p1 »

EXO 5 : JQuery :événementiel

Deuxieme paragraphe

texte du deuxième paragraphe
blabla

Ma belle div

Événement => **mouseleave** : null id =

Coordonnées souris => **MOVE** x= 494 y=385

Élément HTML déclencheur => **click** sur [object HTMLParagraphElement] id = p2

Proposition de corrigés : **exo5.html** (dans **Corriges**, projet NetBeans **CorrigesjQuery**)

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

4. GERER LE DOM ET LES STYLES EN JQUERY

jQuery permet de manipuler HTML et les styles CSS en :

- récupérant et modifiant des attributs d'éléments HTML existants (vu précédemment)
- créant ou supprimant des éléments (à voir)
- récupérant, modifiant, créant ou supprimant des styles CSS (à voir)

4.1 MANIPULER LE DOM EN JQUERY

jQuery possède 4 méthodes pour insérer de nouveaux éléments :

<i>append</i>	insère du contenu à la fin des éléments sélectionnés
<i>prepend</i>	insère du contenu au début des éléments sélectionnés
<i>after</i>	insère du contenu après les éléments sélectionnés
<i>before</i>	insère du contenu avant les éléments sélectionnés

et deux méthodes pour supprimer des éléments :

<i>remove</i>	supprime l'élément sélectionné et ses nœuds enfant
<i>empty</i>	« vide » l'élément sélectionné : supprime uniquement les nœuds enfant en conservant l'élément sélectionné

Attention :

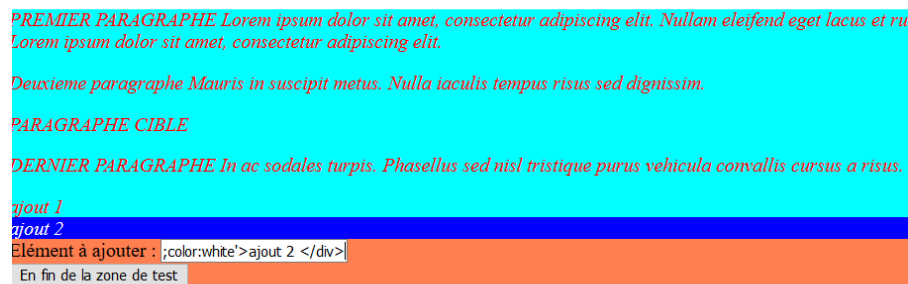
- ces fonctions reçoivent en paramètre du HTML et non du texte : donc éventuellement aussi du JavaScript etc. Pensez à filtrer les données en entrée, pour la sécurité de l'application
- *append* et *prepend* s'appliquent à un ou plusieurs container (*p*, *div*, *span* etc.) qui sont les nœuds parent dans le DOM. *append* ajoute un nouvel élément après le dernier nœud enfant, *prepend* ajoute un nouvel élément avant le premier nœud enfant
- *after* et *before* s'appliquent à un des nœuds enfant : *after* insère le nouvel élément après le nœud enfant, *before* avant le nœud enfant

Lancez la démonstration : **demoDOM.html** (dans **Livrables**, projet NetBeans **TestjQuery**) :
ajoutez des nœuds avec du texte, mais aussi avec du HTML et du JavaScript

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

Jquery : utilisation du DOM



Vous pouvez aussi lancer les démonstrations sur le site [w3schools](https://www.w3schools.com/jquery/jquery_dom_add.asp) :

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

et

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



Exercice 6 :

Complétez le code fourni **demoDOM.html**, en ajoutant les fonctionnalités :

- ajouter un nouvel élément en début de liste
- ... avant un élément de la liste
- ... après un élément de la liste
- supprimer un élément et son arborescence
- vider un élément

Jquery : utilisation du DOM

ajouté au début

PREMIER PARAGRAPHE Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam eleifend eget lacus et rutrum. Phasellus vel faucibus sem, a pellentesque metus. sit amet, consectetur adipiscing elit.

Deuxieme paragraphe Mauris in suscipit metus. Nulla iaculis tempus risus sed dignissim.

ajouté AVANT

PARAGRAPHE CIBLE

ajouté APRES

DERNIER PARAGRAPHE In ac sodales turpis. Phasellus sed nisl tristique purus vehicula convallis cursus a risus. Nunc ornare augue et pharetra faucibus.

ajouté à la fin

Elément à ajouter :

Proposition de corrigés : **exo6.html** (dans **Corriges**, projet NetBeans **CorrigesjQuery**)

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

4.2 MANIPULER LES STYLES CSS EN JQUERY

<https://api.jquery.com/category/manipulation/class-attribute/>

jQuery permet de manipuler facilement les « classes » CSS :

<i>addClass</i>	ajoute une ou plusieurs classes CSS aux éléments sélectionnés https://api.jquery.com/addClass/
<i>removeClass</i>	supprime une ou plusieurs classes CSS des éléments sélectionnés https://api.jquery.com/removeClass/
<i>toggleClass</i>	bascule entre ajout/suppression de classes CSS dans les éléments sélectionnés
<i>css</i>	modifie ou retourne un attribut de style https://api.jquery.com/css/

Vous pouvez aussi utiliser la démonstration de *w3schools* sur l'utilisation des css en jQuery:

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



Exercice 7 :

Vous allez consulter la documentation en ligne de jQuery, sur les classes et les styles.

En adaptant et combinant à votre goût les exemples de la documentation, effectuez :

- un ajout de classe simple sur un ou plusieurs éléments
- un ajout de plusieurs classes sur un ou plusieurs éléments
- une suppression de classe sur un ou plusieurs éléments
- une modification de style (par exemple la taille des fontes) sur tous les éléments.

Proposition de corrigés : **exo7.html** (dans **Corriges**, projet NetBeans **CorrigesjQuery**)

4.3 PARCOURIR L'ARBORESCENCE DU DOCUMENT AVEC JQUERY

4.3.1 Rappels sur le DOM et les parcours d'arbre

jQuery permet de parcourir l'arborescence d'un document (*Traversing*), de manière plus pratique que l'API du DOM (*document.getElementById*, *document.getElementsByTagName* etc.).

Avant de présenter les méthodes de jQuery, il faut rappeler la structure d'un document HTML.

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

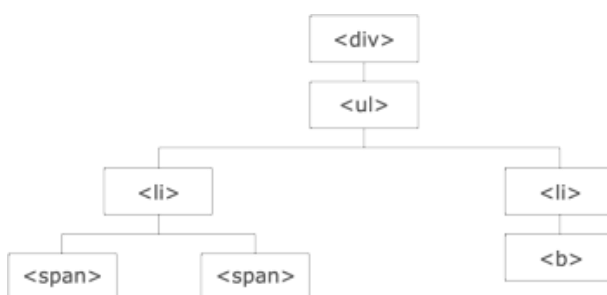
Chaque élément (ou balise, *tag*) HTML est un « *nœud* » du document courant (objet racine *document* dans la bibliothèque DOM en JavaScript).

jQuery permet de :

- remonter d'un nœud vers son parent (ou vers l'un de ses ancêtres)
- lister les enfants d'un nœud donné (ou ses descendants)
- chercher le nœud avant/après un élément, au même niveau de l'arbre (un membre de la même fratrie, en anglais « *siblings* »)

Regardons cet exemple de document HTML (fourni par *w3schools*) :

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



<div> est le **parent** de **** et l'ancêtre de tous les autres nœuds

**** est le **parent** de deux **** et l'enfant (**child**) de **<div>**

Le **** de gauche est le **parent** de deux ****, l'enfant de **** et le **descendant** de **<div>**

Les deux **** sous **** sont « frères » (**siblings**) : ils ont le même parent.

etc.

4.3.2 Mise en pratique

Pour mettre en pratique les méthodes de parcours d'arbre fournies par jQuery, vous allez vous appuyer sur *w3schools* qui est très visuel : *jQuery Traversing*

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

et suivants, jusqu'à *jQuery Filtering*.

4.3.3 Les méthodes de parcours d'arbre

Quelques méthodes jQuery pour parcourir l'arbre HTML et agir sur lui (non exhaustif) :

children retourne la liste des enfants d'un élément HTML (uniquement les descendants directs)

`$("#div1").children();` // tous les enfants de div1

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

```
$("#div1").children("p"); // les paragraphes, enfants de div1
$("#div1").find("p.titre"); // les paragraphes, enfants de div1, qui possèdent la classe titre
```

find retourne la liste de tous les descendants (enfants, petits-enfants etc.) d'un élément, pour le type demandé (ou * pour tous les types)

```
$("#div1").find("*"); // tous les descendants de div1
$("#div1").find("p"); // tous les paragraphes qui descendent de div1
```

first retourne le premier élément d'une liste

```
$("p").first(); // le premier paragraphe
```

last retourne le dernier élément d'une liste

```
$("p").last(); // le dernier paragraphe
```

eq(i) retourne l'élément d'indice i dans une liste (à partir de 0)

```
$("p").eq(1); // le deuxième paragraphe
```

filter(critere) retourne les éléments correspondant au critère

```
$("div").filter(".entete") // toutes les div qui ont la classe entete
```

parent retourne le parent direct du/des élément

```
$("p").parent() // les éléments parent auxquels appartiennent les paragraphes
```

parents retourne tous les ancêtres (parent, grands-parents etc.) de/des éléments

```
$("p").parents() // tous les éléments ancêtres des paragraphes
```

etc.

Vous trouverez le détail de ces méthodes, avec des démonstrations visuelles sur le site de w3schools :

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

5. EXERCICE DE SYNTHÈSE

Pour mettre en pratique jQuery et en particulier les manipulations sur le DOM, nous allons réaliser une **gestion de commandes** simple.

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

5.1 CAHIER DES CHARGES

Nouvelle commande

Id Article	Prix unitaire	Quantité	Sous-Total
peluche1	12.2	5	61
ours2	100	2	200
chatdeco1	200.3	10	2003

Total 2264

Ajouter une ligne

Pour chaque ligne de commande, l'application gère quatre colonnes :

- l'identificateur de l'article
- le prix unitaire de l'article
- la quantité commandée
- le sous-total : prix unitaire * quantité

Et le total général de la commande.

Dans cette maquette simpliste (dont le seul but est de s'entraîner à jQuery) :

- Pas de requêtes vers un serveur SQL ou un web service : tous les champs sont saisis par l'utilisateur
- L'application permet d'ajouter dynamiquement une nouvelle ligne, par le bouton « *Ajouter une ligne* »
- L'utilisateur peut modifier le prix unitaire et la quantité : le sous-total et le total sont recalculés à chaque fois que l'on quitte les champs « Prix unitaire » ou « Quantité »
- Le sous-total et le total ne peuvent pas être modifiés par l'utilisateur
- La dernière ligne ajoutée s'affiche en rouge, les autres en gris
- Le prix est initialisé à 0 et la quantité à 1
- L'application détecte les erreurs de saisie (champ non numérique ou strictement inférieur à 0), et les indique en changeant la couleur de fond du champ
- Les sous-totaux et le total ne doivent pas être faussés par les erreurs de saisie.

Nous allons donner quelques conseils pour construire cette application en jQuery, mais vous restez maîtres de votre code : n'hésitez pas à tester plusieurs façons de faire, pour vous approprier jQuery.

5.2 PREMIERE REGLE : SEPARER HTML ET JAVASCRIPT.

Placez tous les appels jQuery dans un fichier *commande.js*, avec un fichier HTML très simple :

```
<!DOCTYPE html>
<html>
  <head>
```

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

```

<title>Application de commandes</title>
<meta charset="UTF-8">
<script src="jquery-3.4.1.js" type="text/javascript"></script>
<link href="commande.css" rel="stylesheet" type="text/css"/>
<script src="commande.js" type="text/javascript"></script>
</head>
<body>
  <div>
    <h1>Nouvelle commande</h1>
    <table id="idtable">
    </table>

    <p>Total <label id="idtotal">0</label></p>

    <button id="btnAjoutLigne">Ajouter une ligne</button>
  </div>
</body>
</html>

```

Notre travail avec jQuery pour ajouter et modifier les lignes se fera sur la table *idtable*
Avec un total dans le label *idtotal*

5.3 DEUXIEME REGLE : REALISER LES TRAITEMENTS AVEC JQUERY, ETAPE PAR ETAPE

5.3.1 Etape 1

Comme dans les exemples précédents, nous allons placer le code événementiel dans une méthode *initComponents* qui sera appelée à la fin du chargement du document.

Ceci permet de ne pas encombrer le code HTML avec du JavaScript.

Exemple de maquette de départ : fichier *commande.js*

```

// Ajout d'une nouvelle ligne
function onBtnClickAjout()
{
  alert ("click");
}

// initialise tous les composants
function initComponents()
{
  $("#btnAjoutLigne").click(onBtnClickAjout);
}

// appelle la fonction d'initialisation après chargement du document
$(document).ready(initComponents);

```

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Vérifiez que le *click* est bien géré par jQuery :



5.3.2 Ajouter une nouvelle ligne sans présentation

Nouvelle commande

Id Article	Prix unitaire	Quantité	Sous-Total
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Total 0

```
function onBtnClickAjout()
{
    // création et initialisation de la nouvelle ligne
    var ligne = "<td>Id Article <input type='text'/></td>" +
        "<td>Prix unitaire <input type='text'/></td>" +
        "<td>Quantité <input type='number' min='0' class='classqte' /></td>" +
        "<td>Sous-Total <input type='text' class='stotal' readonly /></td>";

    var tr = $("<tr></tr>").html(ligne);
    // ajout de la ligne à la table idtable
    $("#idtable").append(tr);
}
```

On construit le champ HTML de la ligne (une liste de `<td>`) puis on crée l'objet jQuery `<tr>` par la méthode *html*.

On ajoute l'objet jQuery `<tr>` à la table par la méthode *append*

5.3.3 Ajouter les règles de gestion sur les nouvelles lignes

Avant l'ajout de la nouvelle ligne, on remet la ligne précédente en gris.

La ligne précédente, avant l'ajout, est le dernier élément `<tr>`, enfant de la table `#idtable` :

#idtable tr:last

```
$("#idtable tr:last").css({"color": "grey"});
```

Après l'ajout de la nouvelle ligne, on utilisera la même syntaxe pour la passer en rouge.

Pour initialiser les champs prix, quantité et sous-total :

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

- ces champs sont des descendants de la dernière ligne `<tr>`
- mais pas ses enfants, car il y a les nœuds `<td>` entre le nœud `<tr>` et les nœuds `<input>`
- donc on utilisera un *find*, avec un paramètre "input"

```
// récupération de la nouvelle ligne que l'on passe en rouge
```

```
var dernier = $("#idtable tr:last");
```

```
dernier.css({"color": "red"});
```

```
var inputPrix = dernier.find("input").eq(1);
```

```
inputPrix.addClass("classprix");
```

```
inputPrix.attr("value", 0);
```

```
// voir suite
```

La méthode *eq(i)* récupère l'élément numéro i dans une liste : ici le prix

La méthode *addClass* ajoute une classe css à un élément : ici *classprix*

La méthode *attr* permet de modifier un attribut : ici *value* à 0

Nouvelle commande

Id Article	Prix unitaire	Quantité	Sous-Total
<input type="text"/>	<input type="text" value="0"/>	<input type="text" value="1"/>	<input type="text" value="0"/>
Id Article	Prix unitaire	Quantité	Sous-Total
<input type="text"/>	<input type="text" value="0"/>	<input type="text" value="1"/>	<input type="text" value="0"/>
Id Article	Prix unitaire	Quantité	Sous-Total
<input type="text"/>	<input type="text" value="0"/>	<input type="text" value="1"/>	<input type="text" value="0"/>

Total 0

[Ajouter une ligne](#)

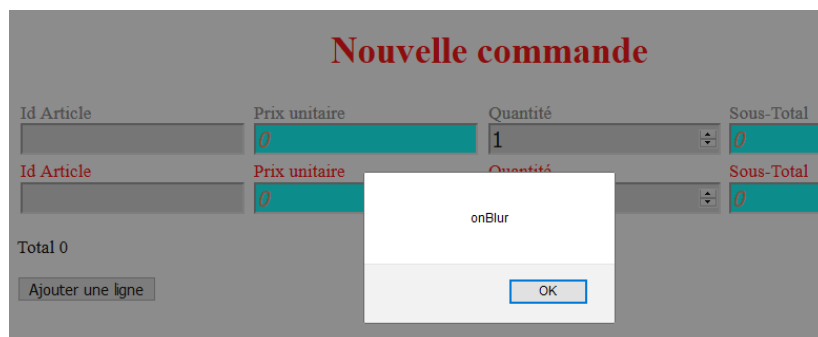
5.3.4 Ajouter la gestion des modifications de prix et de quantité

Sur les champs *Prix* et *Quantité*, on associe la méthode *onBlur* à l'événement *blur*, pour actualiser le sous-total et le total, dès que l'on quitte un de ces champs :

```
// suite
```

```
inputPrix.blur(onBlur);
```

Commencez par tester la méthode *onBlur* avec *alert()*



Il faut réfléchir au parcours à effectuer dans le DOM, pour mettre à jour le sous-total et le total.

Le paramètre *event* de la méthode *onBlur* référence l'élément que l'on vient de quitter :

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

`event.currentTarget`

Cet élément est un `<input>`, qui est enfant d'un élément `<td>` et le descendant de la ligne `<tr>`.

Mais attention : c'est un objet du DOM (JavaScript) et pas un objet jQuery.

Il faut donc commencer par construire un objet jQuery, avec la syntaxe habituelle :

`$(event.currentTarget)`

Pour trouver les autres éléments `<input>` de la ligne, le plus pratique est de remonter tous les ascendants, par la méthode `parents`, jusqu'au `<tr>` :

`$(event.currentTarget).parents("tr");`

Pour vérifier le parcours d'arbre, écrivez des valeurs bidon dans les champs `<input>` :

Nouvelle commande

<small>Id Article</small>	<small>Prix unitaire</small>	<small>Quantité</small>	<small>Sous-Total</small>
	<input type="text" value="=> prix"/>	<input type="text" value="100"/>	<input type="text" value="=> sous total"/>

Total 0

Ajouter une ligne

```
function onBlur(event)
{
    var ligne = $(event.currentTarget).parents("tr");
    var inputs = ligne.find("input");
    inputs.eq(1).val("=> prix") ;
    inputs.eq(2).val(100) ;
    inputs.eq(3).val("=> sous total");
}
```

Il reste à :

- récupérer les valeurs des champs
- vérifier qu'elles sont numériques
- faire le calcul et mettre à jour le sous-total, ou appliquer une classe css au champ en erreur.

Nouvelle commande

<small>Id Article</small>	<small>Prix unitaire</small>	<small>Quantité</small>	<small>Sous-Total</small>
ours1	<input type="text" value="12"/>	<input type="text" value="10"/>	<input type="text" value="120"/>
Id Article	<small>Prix unitaire</small>	<small>Quantité</small>	<small>Sous-Total</small>
peluche2	<input type="text" value="13"/>	<input type="text" value="greu!"/>	<input type="text" value="130"/>

Total 0

Ajouter une ligne

```
function onBlur(event)
{
```

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery


```

var ligne = $(event.currentTarget).parents("tr");
var inputs = ligne.find("input");

var prix = parseFloat(inputs.eq(1).val() );
var qte = parseInt(inputs.eq(2).val() );

if (Number.isNaN(qte) || Number.isNaN(prix) || prix < 0 || qte < 0)
{
    $(this).addClass("erreur");
}
else
{
    $(this).removeClass("erreur");
    inputs.eq(3).val(qte * prix);
}
}

```

Si les champs *Prix* ou *Quantité* ne sont pas numériques, la conversion en JavaScript renvoie "**NaN**" (*Not a Numeric*) qui peut être testé par la fonction booléenne *Number.isNaN*

5.3.5 Mettre à jour le total

Pour parcourir les sous-totaux et recalculer le nouveau total en jQuery, il est pratique de :

- marquer les sous-totaux avec une classe css pour les retrouver facilement dans la table : *stotal*
- utiliser la méthode *each* sur la collection retournée par le *find*, avec une fonction anonyme passée en paramètre, pour faire le calcul :

```

function majTotal()
{
    var total = 0;
    $("#idtable").find("input.stotal").each(
        function ()
        {
            total += parseFloat($(this).val());
        });

    $("#idtotal").text(total);
}

```

Proposition de corrigés : **commande.html** (dans **Corriges**, projet NetBeans **CorrigesjQuery**)

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

6. UTILISER AJAX AVEC JQUERY

D'abord un résumé rapide :

AJAX (*Asynchronous JavaScript and XML*) n'est ni un langage ni une technique unique, mais un ensemble de techniques Web qui permettent de charger des contenus en asynchrone, et de les greffer dans la page courante, sans avoir à changer de page.

jQuery fournit plusieurs méthodes AJAX, permettant de charger du texte brut, HTML ou JSON, et de les insérer dans un élément HTML de la page courante.

[https://fr.wikipedia.org/wiki/Ajax_\(informatique\)](https://fr.wikipedia.org/wiki/Ajax_(informatique))

6.1 EXEMPLE 1 : AVEC UN FICHIER LOCAL, PAR LA METHODE LOAD

<https://api.jquery.com/load/>

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

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<meta charset="UTF-8">
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("#divAjax").load("contenu.txt");
    });
  });
</script>
</head>

<body>
  <div id="divAjax"><p>données à charger par un appel AJAX<p></div>
  <button>Charger les données par AJAX</button>
</body>
</html>
```

La méthode *load* appliquée à un élément HTML (ici *#divAjax*) permet d'y insérer un contenu local ou distant, désigné par son url (ici le fichier local "contenu.txt").

Par défaut, cette méthode est synchrone.

Lancez cet exemple : **ajax1.html (Livrables, projet TestjQuery)** et suivez l'appel AJAX avec l'outil *Développement web / Réseau* sous *Firefox* ou son équivalent dans un autre navigateur.

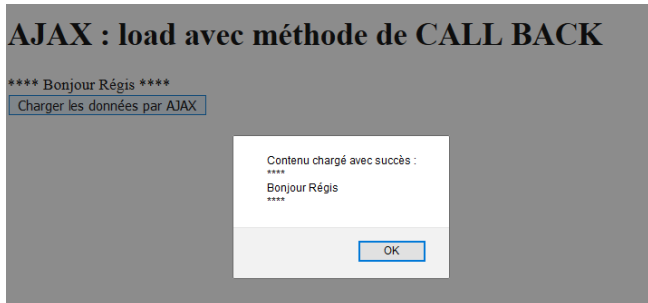
Changez le nom du fichier à charger pour provoquer une erreur : le programme reste muet !

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Afpa © 2019 – Section Tertiaire Informatique – Filière « Etude et développement »

6.2 EXEMPLE 2 : LE MEME EXEMPLE AVEC UNE METHODE DE CALLBACK

Comme en AJAX classique en JavaScript, on peut lancer une méthode événementielle qui s'exécute à la fin du chargement (« *callback method* »), pour tester l'échec ou le succès de l'opération.



```
<script>
$(document).ready(function () {
    $("button").click(function () {
        $("#divAjax").load("contenu.txt", function (responseTxt, statusTxt, xhr)
        {
            if (statusTxt === "success")
            {
                alert("Contenu chargé avec succès : " + responseTxt);
            }
            else if (statusTxt === "error")
            {
                alert("### Erreur de chargement ###" );
            }
        }
    });
});
</script>
```

Le paramètre *responseTxt* contient le document envoyé par le serveur (texte brut, HTML), le paramètre *statusTxt* contient le compte-rendu du serveur (*success* ou *error*).

Le paramètre *xhr* (XMLHttpRequest) est l'objet JavaScript qui permet de manipuler AJAX.

https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax_load_callback

Changez le nom du fichier à charger et vérifiez que le message « Erreur de chargement » s'affiche.

6.3 EXEMPLE 3 : UN APPEL ASYNCHRONE AVEC UNE METHODE DE CALLBACK

Dans les deux premiers exemples, la méthode *load* s'exécute de façon synchrone.

Pour utiliser pleinement les possibilités d'AJAX (***Asynchronous JavaScript and XML***), on peut utiliser la méthode *ajax* de jQuery :

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

<https://api.jquery.com/jquery.ajax/>

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

```
<script>
$(document).ready(function () {
    $("button").click(function () {
        $.ajax( { url: "contenu.txt",
                success: function (result) {
                    $("#divAjax").html(result);
                },
                error : function (jqXHR) {
                    alert (jqXHR.statusText);
                }
            });
    });
});
</script>
```

La méthode *ajax* reçoit en paramètre un objet JSON avec les attributs :

- *url* : url à charger
- *success* : fonction de *callback* à appeler, à l'issue d'un chargement réussi
- *error* : fonction de *callback* à appeler en cas d'échec du chargement.

Changez le nom de fichier à charger et vérifiez que le message d'erreur s'affiche.

6.4 EXEMPLE 4 : RECUPERER DU JSON

La méthode *getJSON* reçoit en paramètres l'url, et une méthode de Callback qui va recevoir et exploiter le résultat JSON :

```
<script>
$(document).ready(function () {
    $("button").click(function () {
        $.getJSON("contenuJson.js", function (result) {
            $.each(result, function (nom, valeur) {
                $("#divAjax").append("<p>" + nom + ": " + valeur + "</p>");
            });
        });
    });
});
</script>
```

Dans cet exemple, le résultat JSON est récupéré dans l'objet *result* qui est parcouru champ par champ par la méthode *each* de jQuery.

Pour chaque attribut de l'objet *result*, la méthode *each* appelle une méthode anonyme passée en paramètre, qui affiche le nom et la valeur de l'attribut.

Ecrire un script JavaScript en utilisant les fonctions de base de jQuery

Vous trouverez les exemples dans **Livrables**, projet **TestjQuery** :
ajax1.html à **ajax4.html**

6.5 POUR ALLER PLUS LOIN AVEC AJAX EN JQUERY

Vous pouvez suivre les démonstrations en ligne du site *w3schools* :
https://www.w3schools.com/jquery/jquery_ref_ajax.asp

CRÉDITS

OEUVRE COLLECTIVE DE L'AFPA

Sous le pilotage de la DIIP
et du centre sectoriel Tertiaire

EQUIPE DE CONCEPTION

Régis Lécu – Formateur AFPA Pont de Claix

Reproduction interdite

Article L 122-4 du code de la propriété intellectuelle.

« Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droits ou ayants cause est illicite. Il en est de même pour la traduction, l'adaptation ou la reproduction par un art ou un procédé quelconque. »