

TP - Interactions dans une page web

M. GEORGES-SAINT-MARC - NSI - Lycée Mendès France Rennes

25 novembre 2022

L'objectif des prochains TP concernant le web est de manipuler des langages permettant :

- de rendre des pages web **dynamiques** (actions sur la page via javascript) : actions côté **client**.
- de créer des pages au **contenu personnalisé** (contenu créé via php) : actions côté **serveur**.

Ce premier TP consacré au Javascript va consister à créer des interactions dans une page web constitué d'un formulaire de connexion et d'un formulaire de commande de produits informatiques.

1 Principe

Le langage qui gère nativement les interactions dans une page web est le **javascript** (créé en 1995 par l'américain Brendan Eich).

C'est un langage **interprété** par le navigateur (tout comme html) et **orienté objet** : nous allons modifier des propriétés et exécuter des fonctions (appelées méthodes) qui sont liées aux différents objets d'une page web.

La syntaxe de ce langage est parfois un peu complexe et des simplifications importantes ont été apportées par la mise en place d'une bibliothèque très courante appelée jQuery. Nous ferons donc appel à cette bibliothèque dans ce TP.

→ Pour cela, on a ajouté à la fin du code html de la page (juste avant la balise `</body>`) la ligne suivante :

```
<script type="text/javascript" src="jquery-3.6.1.min.js"></script>
```

2 Formulaire de connexion

→ Ouvrir la page *formulaire.html* avec un navigateur Web et avec un éditeur de texte (Atom de préférence).

2.1 Utilisation de jQuery

Pour sélectionner un ou plusieurs objet(s) de la page via jQuery, on utilise la syntaxe suivante :

`$("sélecteur élément")`

où *sélecteur élément* est un descriptif du ou des objets. C'est le même type de descriptif qu'en CSS.

Par exemple :

- `$("body")` sélectionne la balise `body`
- `$("td")` sélectionne les balises **td** de la page.
- `$(".Super")` sélectionne les balises de la page ayant la classe *Super*.
- `$("#Unique")` sélectionne la balise de la page ayant l'identifiant *Unique*.

Une fois un objet sélectionné, on peut appliquer des méthodes (des fonctions) qui sont prévues pour ce type d'objet.

Voici quelques méthodes classiques et leur utilité :

<code>\$(objet).show()</code>	Fait apparaître un objet.
<code>\$(objet).hide()</code>	Fait disparaître un objet.
<code>\$(objet).css("propriété","valeur")</code>	Modifie une <i>propriété</i> CSS d'un objet en lui donnant une nouvelle <i>valeur</i> .
<code>\$(objet).val()</code>	Récupère la valeur contenue dans l'attribut value d'un objet.
<code>\$(objet).val(valeur)</code>	Attribue une <i>valeur</i> à l'attribut value d'un objet.
<code>\$(objet).html()</code>	Récupère le contenu html d'un objet.
<code>\$(objet).html(code)</code>	Insère du <i>code</i> html dans un objet.
<code>\$(objet).click(fonction)</code>	Exécute une fonction lors du clic sur un objet.
<code>\$(objet).focus(fonction)</code>	Exécute une fonction lors du focus sur un objet.
<code>\$(objet).blur(fonction)</code>	Exécute une fonction lors du blur sur un objet.

Remarques :


- **Focus** : "arrivée" sur l'objet via le clavier ou la souris (important pour une zone de texte).
- **Blur** : "départ" de l'objet via le clavier ou la souris (important pour une zone de texte).
- **chaque instruction javascript se termine par un point-virgule.**
- Une ligne de commentaire en javascript commence par // ou #.

2.2 Tests via la console JavaScript

Il est possible de mesurer les possibilités du langage javascript en se servant de la **console Javascript** du navigateur.

Exercice 1 :

On suppose que le navigateur Firefox est ouvert à la page *formulaire.html*.

- Ouvrir la console :
 - Sous **Firefox**, appuyer sur **Shift+F5** pour faire apparaître les outils de développement.
 - Puis sélectionner l'**onglet Console** (s'il n'apparaît pas, utiliser le bouton  pour le faire apparaître).
- La console fonctionne comme pour Python. On y tape des commandes qui s'exécutent immédiatement. Saisir la commande suivante à la suite du symbole `>>` :

```
1 >> $( "#Connexion" ).hide ()
```

Que s'est-il passé ?

.....

.....

- Rafraîchir la page (avec la touche F5). Que se passe-t-il pour le formulaire de connexion ?
-

- Dans la console, taper uniquement `$("#Connexion")` puis appuyer sur *Entrée*.
Une ligne avec `Object { 0: section#Connexion, length: 1 }` doit apparaître, précédée du triangle ▷.
 - Cliquer sur le triangle, puis sur le triangle situé devant `<section id="Connexion">` . On retrouve toutes les caractéristiques et fonctions liées à l'objet `$("#Connexion")` . Trouver parmi tous ces éléments le nom de la caractéristique qui indique quelle est la balise qui **contient** la balise `#Connexion` (*son noeud parent*).

.....

- Quel objet atteint-on en saisissant dans la console la commande suivante ?

```
1 $( "#Connexion" ) [0].firstElementChild
```

.....

2.3 Au travail !

Passons à la manipulation durable de la page.

Le formulaire affiche plusieurs données qui s'avèrent inutiles par défaut sur la page. Il convient donc de les cacher par défaut et de ne les faire apparaître que si elles deviennent utiles.

Pour les questions suivantes, on écrira le code javascript entre les balises **<script>** et **</script>** qui suivent la ligne faisant appel à jQuery (ligne 111 du fichier).

1. Le document est découpé en deux sections : #Connexion et #Commande. Cacher la section #Commande au chargement de la page à l'aide d'une commande javascript.
.....
2. La section #Connexion n'est pas très mise en relief. À l'aide d'une commande javascript, changer la propriété CSS adéquate pour que l'encadré de connexion soit sur fond coloré (vous êtes libre de la couleur).
.....
3. On souhaite mettre une bordure colorée à l'encadré de connexion lorsque l'utilisateur se focalise dessus. Pour cela, on va construire une **fonction** javascript qui sera appelée au focus sur l'objet. La syntaxe d'une fonction est la suivante en javascript :

```

1  function nomFonction (parametres)
2  {
3      Lignes de code;
4  }
```

- (a) Créer une fonction **bordureConnexion** sans paramètre qui change la couleur de la bordure de #Connexion en rouge.

- (b) On fera appel à cette fonction pour le focus sur #Login avec la commande :

```

1  $( "#Login" ).focus (bordureConnexion) ;
```

4. Appeler aussi cette fonction lors d'un focus sur #Password.
5. Créer une fonction **alerteLogin** qui vérifie si le login n'est pas une chaîne vide "" et qui envoie une alerte si c'est le cas avec l'instruction `alert("message")`. Pour écrire une instruction conditionnelle, voici la structure :

```

1  if (condition)
2  {
3      // Instructions
4  }
5  else
6  {
7      // Autres instructions
8  }
```

→ *Indication* : on récupérera l'*identifiant* saisi par l'utilisateur à l'aide de la commande `.val()`.

- Appeler la fonction **alerteLogin** lors du blur de l'input **#Login**.
- Créer une fonction **validerConnexion** qui vérifie si le Login **et** le mot de passe sont bien remplis (donc non vides) et l'appeler lors du clic sur le bouton **#VerifierConnexion**.

Indication : pour vérifier deux conditions simultanément (**and**), on fait :

```
1  if (condition1 && condition 2)
```

Pour ne vérifier qu'au moins une des conditions (**or**), on aura les symboles `||`.

2.4 Transition

- Améliorer la fonction **validerConnexion** en faisant apparaître la section **#Commande** et disparaître la section **#Connexion** si le login et le mot de passe ont bien été complétés.
→ *Remarque* : on pourra utiliser les fonctions **slideDown** et **slideUp** pour gérer ces changements d'états avec des effets.
- Lors de la validation, ajouter aussi l'effet suivant : Insérer le texte "Bonjour " + *identifiant* dans le paragraphe **#Bonjour** à l'aide de la bonne commande javascript.

3 Commande

Dans cette deuxième partie, on veut créer des fonctions de calcul pour la commande des produits.

- Créer une fonction qui permet d'afficher dans **#TotalCarteMere** le total correspondant à la commande de cartes-mères.
→ *Indications* :
 - on fera appel à cette fonction après un changement sur l'input de quantité : fonction `.change()`;
 - on forcera la conversion en nombre entier pour la valeur récupérée dans l'input *CarteMere* avec la fonction **parseInt(valeur)**.
- Faire de même pour chaque item de la commande, en ajoutant les bons identifiants dans le fichier html.
- Créer la fonction permettant de faire le calcul total.
- Améliorer l'ensemble en utilisant les checkbox pour les calculs. On pourra vérifier si un checkbox est coché en faisant :

```
1  if ( $(objet).is(" :checked") )
2  {
3      // instructions
4  }
```