Formulaires HTML5, JavaScript pour vérifier et interagir

Pour permettre à l'utilisateur de communiquer des informations à partir d'une page web, on utilise principalement des formulaires HTML. Les informatisations contenues dans ces formulaires, une fois soumises (lorsque l'utilisateur valide le formulaire), peuvent être traitées par le serveur (en PHP et MySQL par exemple). Mais il est aussi possible de traiter ces informations en JavaScript, c'est-à-dire directement dans le navigateur web, principalement pour vérifier que les informations fournies correspondent à ce qui est demandé. On parle alors de vérification ou de validation des données des formulaires.

I. Sommaire

II. Retour sur Bootstrap et les formulaires HTML

II.1. Rappels avec un exemple

En HTML on décrit un formulaire en le plaçant dans un élément **<form></form>** qui va contenir des contrôleurs ou *champs* que l'utilisateur pourra remplir, et un ou des *boutons* de validation.

Voici un exemple de formulaire que nous allons commenter :

```
<div class="container">
   <form id="formulaire_test" class="form-horizontal" action="/ma-page-de-traitement"
 method="post">
   <fieldset>
    <legend>Qui êtes vous ?</legend>
    <div class="form-group ">
       <label for="choix_nom" class="control-label">Nom :</label>
       <input type="text" id="choix_nom" name="nom" />
    </div>
   </fieldset>
   <fieldset>
    <legend>Quel est votre genre</legend>
    <div class="form-group ">
     <input type="radio" id="choix_femme" name="genre" value="genre_femme" />
     <label for="choix_femme" class="control-label">Femme</label>
    </div>
    <div class="form-group ">
     <input type="radio" id="choix_homme" name="genre" value="genre_homme" />
     <label for="choix_homme" class="control-label">Homme</label>
    </div>
    <div class="form-group ">
     <input type="radio" id="choix_mauvais" name="genre" value="genre_mauvais" />
     <label for="choix_mauvais" class="control-label">Mauvais</label>
    </div>
    <div class="form-group ">
     <input type="radio" id="choix_autre" name="genre" value="genre_autre" />
     <label for="choix_autre" class="control-label">Autre</label>
    </div>
   </fieldset>

    <div class="buttons">
       <button type="submit" value="ok" class="btn btn-primary btn-lg
 active">Confirmer</button>
       <button type="submit" value="nok" class="btn btn-default btn-lg
 active">Annuler</button>
    </div>
  </form>
</div>
```

La Figure 1 ci-dessous montre le résultat de ce code HTML (inséré dans une base Bootstrap).

Figure 1 : Capture du formulaire dont le code HTML est donné ci-dessus

Les balises **<form></form>** prennent des attributs qui déterminent la façon dont le formulaire sera traité du côté du serveur (par exemple en PHP). Voir le cours « *Notion 2 - Php* ».

Il existe différents types de champs que l'on peut donner à remplir aux visiteurs. Les principaux sont :

- **<input />** est le plus utilisé. Selon son attribut type, il peut servir à :
  - type="text" faire remplir un texte court (cf. l'exemple de formulaire ci-dessus).
  - type="radio" faire choisir *entre* différentes options (cf. l'exemple de formulaire ci-dessus). Leur attribut value contient *la* valeur qui sera renvoyée par le formulaire lorsqu'ils ont été cochés.
  - type="checkbox" faire choisir *une ou plusieurs parmi* différentes options (se présente sous la forme de cases à cocher). Leur attribut value contient *une des* valeurs qui sera renvoyée par le formulaire lorsqu'ils ont été cochés (notez que si les boutons radio ne renvoient qu'une seule valeur, les checkboxes peuvent renvoyer plusieurs valeurs).
  - type="submit" faire soumettre le formulaire (ce champ prend en fait la forme d'un bouton) son attribut value détermine l'information qui sera envoyée. On leur préfèrera souvent les balises **<button></button>** plus faciles à styler avec des CSS.
- la combinaison **<select></select>** et **<option></option>** fait choisir dans un menu déroulant, soit entre des options, soit une ou plusieurs options parmi un choix (selon la présence de l'attribut multiple="multiple")
- **<textarea></textarea>** permet de faire remplir un texte long aux utilisateurs.

Tous ces champs sont accompagnés d'un attribut name qui permet de les identifier lors de l'envoi des données. Ils peuvent aussi avoir des champs id et class pour, notamment, le ciblage CSS.

Pour la validation (c'est-à-dire la soumission) des formulaires, on peut utiliser un **<input** type="submit" value="Confirmer" **/>** ou plutôt, comme dans l'exemple de code donné ci-dessus, un **<button** type="submit" value="ok"**>**Confirmer**</button>** qui est plus facile à styler en CSS et permet de différencier la valeur renvoyée (attribut value) et le texte affiché dans le bouton (placé dans la balise). On remarque dans l'exemple que l'on peut mettre plusieurs boutons de soumission pour un même formulaire, c'est alors à vous de gérer les effets en les différenciant grâce à la valeur de l'attribut value renvoyé.

II.2. Quelques points d'accessibilité des formulaires

Pour rendre les formulaires HTML facilement utilisables par des navigateurs classiques et les navigateurs non visuels (navigateurs utilisés par les malvoyants ou robots d'indexation des sites), il faut respecter quelques règles.

II.2.1. Labelliser les champs

Les champs ne doivent pas figurer seuls dans un formulaire, mais être accompagnés d'un **<label></label>** qui les décrit. Le label porte un attribut for qui doit pointer vers l'attribut id du champ auquel il correspond. Exemple issu du code proposé ci-dessus :

```
<input type="radio" id="choix_femme" name="genre" value="genre_femme" />
<label for="choix_femme" class="control-label">Femme</label>
```

Le label permet d'indiquer le texte « Femme » à côté du bouton (aussi bien sur les navigateurs classiques que sur les navigateurs non visuels ou, par exemple, ce texte pourra être lu). De plus, le label permet de faciliter le clic, puisqu'un clic sur le label équivaut à un clic sur le bouton (plus difficile à atteindre). Il en est de même pour les checkboxes.

II.2.2. Regrouper les champs

Pour faciliter la compréhension de vos formulaires, il faut regrouper les champs dans des balises **<fieldset></fieldset>** dont le contenu doit commencer par un **<legend></legend>** qui décrit ce regroupement. Exemple issu du code proposé ci-dessus :

```
<fieldset>
 <legend>Quel est votre genre</legend>
 <div class="form-group ">
  <input type="radio" id="choix_femme" name="genre" value="genre_femme" />
  <label for="choix_femme" class="control-label">Femme</label>
 </div>
 <div class="form-group ">
  <input type="radio" id="choix_homme" name="genre" value="genre_homme" />
  <label for="choix_homme" class="control-label">Homme</label>
 </div>
 <div class="form-group ">
  <input type="radio" id="choix_mauvais" name="genre" value="genre_mauvais" />
  <label for="choix_mauvais" class="control-label">Mauvais</label>
 </div>
 <div class="form-group ">
  <input type="radio" id="choix_autre" name="genre" value="genre_autre" />
  <label for="choix_autre" class="control-label">Autre</label>
 </div>
</fieldset>
```

On voit ici que le <**fieldset**></**fieldset**> permet de regrouper les boutons radios qui portent sur la même question.

II.3. Bootstrap et les formulaires

Le framework CSS Bootstrap facilite grandement la mise en place des formulaires. Il permet en premier lieu de donner une présentation au formulaire qui les rend très lisibles (grâce aux CSS fournies). Selon les cas, vous aurez néanmoins des modifications de positionnement à opérer pour améliorer encore la lisibilité. Pour cela vous pourrez utiliser les classes classiques de placement sur la grille Bootstrap.

En second lieu, il apporte quelques classes spécifiques aux formulaires dont des classes permettant de donner un style particulier aux boutons principaux et secondaires ce qui est une bonne pratique (voir la Figure 1).

```
        <button type="submit" value="ok" class="btn btn-primary btn-lg
active">Confirmer</button>
        <button type="submit" value="nok" class="btn btn-default btn-lg
active">Annuler</button>
```

La classe btn désigne les boutons, la classe btn-primary et btn-default permet de les présenter différemment, tandis que la classe active permet de montrer que les boutons sont actifs. Voir : http://getbootstrap.com/css/#buttons

Ainsi que des classes facilitant la mise en forme des formulaires comme form-horizontal pour le formulaire lui-même, form-group pour l'élément qui regroupe le label et le champ, control-label pour les labels, etc. (voir http://getbootstrap.com/css/#forms).

II.4. Pour compléter à propos des formulaires HTML et Bootstrap

Voici quelques liens pour aller plus loin :

- documentation sur les formulaires HTML : https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires
- documentation sur l'accessibilité des formulaires : http://openweb.eu.org/articles/formulaire_accessible
- Les classes Bootstrap pour les formulaires : http://getbootstrap.com/css/#forms

II.5. Exercice 17

Réalisez un formulaire de contact pour le CV (à placer, par exemple, sous le reste du CV). Ce formulaire de contact doit contenir (voir aussi la Figure 2) :

- un champ pour rentrer son nom
- un champ pour entrer son email
- un champ « thème » qui permet de choisir un sujet dans un menu déroulant. Voici quelques propositions de sujets (vous êtes libre de mettre ceux que vous voulez) :
  - Je vous contacte à propos de vos travaux de recherche
  - Je vous contacte à propos d'un emploi
  - Je vous contacte, car je suis une ancienne connaissance
  - Autre raison
- un champ pour le contenu du message
- un champ antispam avec des boutons radio pour vérifier que l'utilisateur n'est pas un robot de spam
- un bouton d'envoi du formulaire.

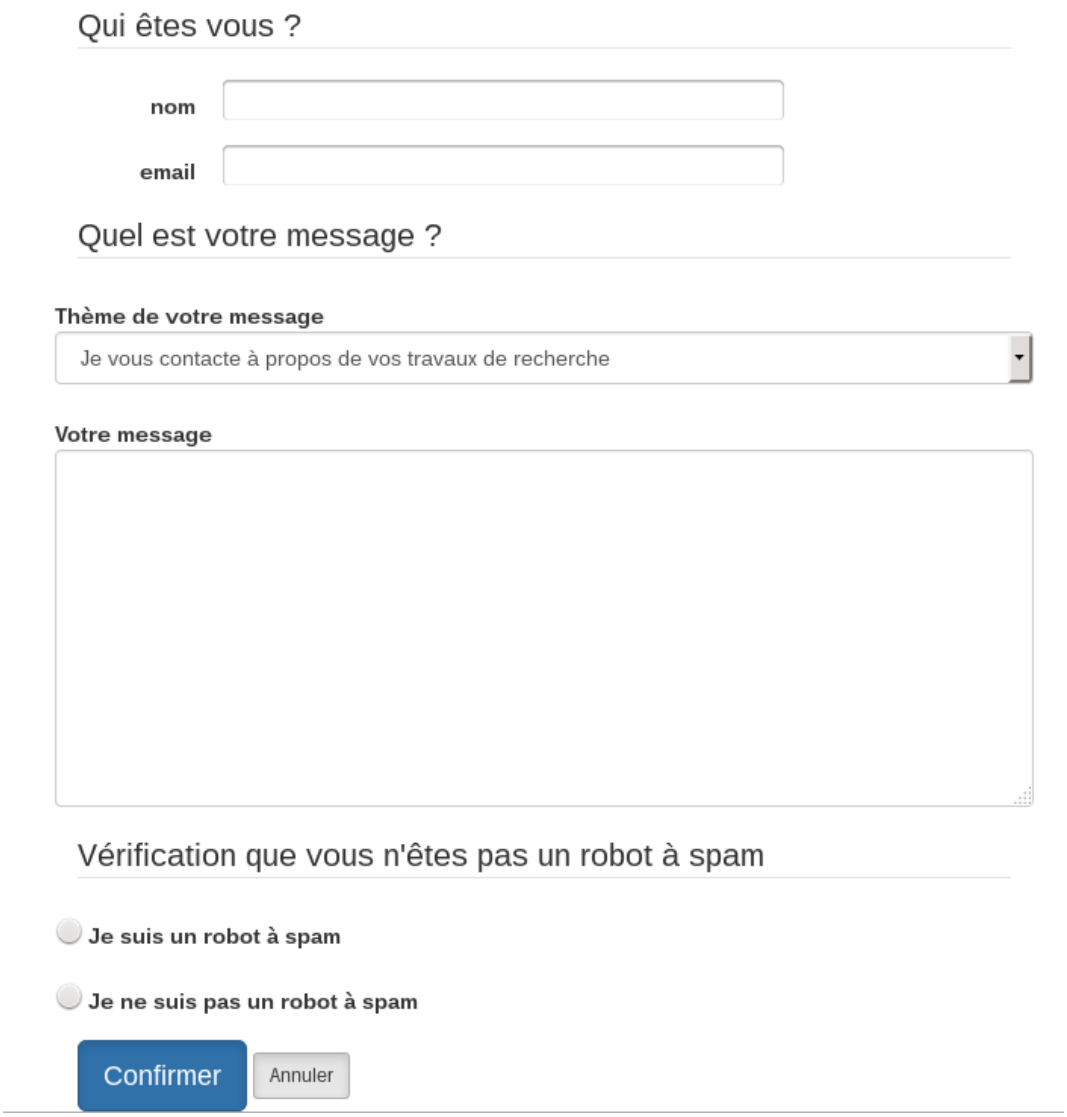

Figure 2 : Capture d'un exemple de formulaire pour l'exercice 17

III. HTML5 : de la suggestion à l'auto-vérification

Grâce au HTML5, il est possible d'aller plus loin dans l'aide apportée aux utilisateurs des formulaires.

III.1. La suggestion par le placeholder

Pour guider les utilisateurs, il est possible de montrer un exemple en pré-remplissant certains champs grâce à l'attribut placeholder. Ex :

```
<label for="choix_nom" class="control-label">Nom :</label>
<input type="text" id="nom" placeholder="ex. : Jeanne Dupond"/>
```

La Figure 3 ci-dessous montre une capture du résultat dans le navigateur de ce code HTML (placé dans une base Bootstrap) :

Figure 3 : Exemple de placeholder

On peut placer des attributs placeholder sur les champs **<input />** de type text, search, password, url, tel et email et sur les **<textarea></textarea>**.

Voir aussi : http://www.alsacreations.com/tuto/lire/1370-formulaire-html5-placeholder-required-pattern.html

III.2. La vérification simple avec l'attribut required

L'attribut required permet de vérifier que l'utilisateur a bien rempli un champ obligatoire : lorsque qu'un champ a l'attribut required, le navigateur vérifie après validation que ce champ est rempli, s'il ne l'est pas, il n'envoie pas le formulaire et affiche un message. Ex. :

```
<label for="choix_nom" class="control-label">Nom :</label>
<input type="text" id="nom" placeholder="ex. : Jeanne Dupond" required="required"/>
<em>Obligatoire !</em>
```

Avec ce code HTML, si le champ n'est pas rempli et que le formulaire est validé, le résultat sera :

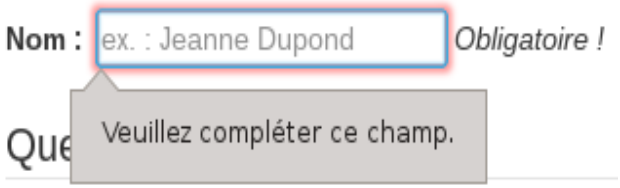

Figure 4 : Exemple de retour après auto-vérification d'un champ portant l'attribut required

Attention, il convient aussi de prévenir l'utilisateur qu'un champ est obligatoire. Ici c'est fait grâce au ****Obligatoire !****.

Voir aussi : http://www.alsacreations.com/tuto/lire/1391-formulaire-html5-placeholder-required-pattern.html

III.3. L'auto-vérification avancée avec les motifs (patterns)

Pour aller plus loin dans l'auto-vérification des formulaires par le navigateur, il est possible de vérifier que le contenu des informations données par l'utilisateur correspondent à un motif prédéfini. Cette auto-vérification utilise l'attribut pattern qui doit contenir une expression régulière (à ce sujet, voir par exemple http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/les-expressions-regulieres-partie-1-2). Voici un exemple de formulaire permettant d'entrer sa date préférée et qui vérifie que l'entrée de l'utilisateur est conforme au format JJ/MM/YYYY.

```
<fieldset>
    <legend>Votre date préférée ?</legend>
    <div class="form-group ">
      <label for="choix_date" class="control-label">Date (<em>sous la forme
JJ/MM/YYYY</em>) :</label>
      <input
        type="text"
        id="choix_date"
        placeholder="ex. : 01/01/1970"
        pattern="(0[1-9]|1[0-9]|2[0-9]|3[01])[- /.](0[1-9]|1[012])[- /.][0-9]{4}" />
    </div>
  </fieldset>
```

La Figure 5 ci-dessous montre l'affichage lorsque l'on tente de valider le formulaire avec une date ne correspondant pas au motif.

Figure 5 : Exemple de retour donné à l'utilisateur lorsque l'auto-validation d'un formulaire montre que les données ne correspondent pas au motif.

À noter qu'il existe des dérivés des champs **<input** type="text" **/>** qui proposent des motifs intégrés. C'est le cas des types tel, url, email... Mais, même dans ces cas-là, l'ajout d'un pattern peut s'avérer nécessaire. À noter aussi que les patterns ne s'appliquent pas aux **<textarea></textarea>**.Voir : http://www.alsacreations.com/tuto/lire/1403-formulaire-html5-type-tel.html (et suivantes).

Pour en savoir plus :

- Sur l'attribut pattern : http://www.alsacreations.com/tuto/lire/1392-formulaire-html5-placeholder-required-pattern.html
- Sur les exemples de patterns possibles : http://html5pattern.com/

III.4. Exercice 18

Modifiez le formulaire de l'exercice 17 de façon à :

- indiquer qu'ajouter un email et choisir entre les boutons radios antispam est obligatoire (avec auto-vérification)
- guider l'utilisateur sur ce qu'il doit entrer dans le champ email et dans le champ du texte du message
- auto-vérifier que l'utilisateur met bien une adresse email dans le champ email et que le nom contient au moins 3 caractères.

III.5. Plus loin avec l'accessibilité : les attributs ARIA

Pour ceux qui souhaitent aller plus loin dans les retours proposés aux utilisateurs, notamment ceux qui n'utilisent pas des navigateurs classiques, vous pouvez explorer la piste des attributs ARIA :

- http://fr.clever-age.com/veille/blog/ameliorer-l-accessibilite-d-un-formulaire.html
- https://developer.mozilla.org/fr/docs/Accessibilit%C3%A9/ARIA/formulaires/Indications_elementaires_pour_les_formulaires

- http://www.lesintegristes.net/2008/12/09/introduction-a-wai-aria-traduction/

IV. Vérifier et dynamiser un formulaire avec JavaScript et jQuery

Les auto-vérifications proposées par les attributs dédiés du HTML5 sont pratiques, mais pas toujours suffisantes. Notamment, elles ne permettent pas de :

- Conseiller l'utilisateur en fonction de ses erreurs de saisie (ex. : sur la Figure 5 indiquer à l'utilisateur qu'il a fait une erreur sur le mois).
- Modifier le formulaire pour mettre en évidence les erreurs (ex. : ajouter des exemples d'entrées valides)
- Interagir avec l'utilisateur au fur et à mesure de sa frappe (ex. : montrer interactivement que ce qui est saisi est correct)
- etc.

D'autre part, le JavaScript permet d'enrichir les interactions avec l'utilisateur en ajoutant, retirant ou modifiant des éléments de formulaires en fonction de ses choix précédents.

IV.1. Des événements liés aux formulaires

jQuery propose des événements qui permettent d'interagir avec les utilisateurs d'un formulaire.

IV.1.1. Submit

La méthode submit de jQuery permet de réagir lorsque le formulaire est soumis par l'utilisateur (c'est-à-dire lorsqu'il clique sur un des **<button></button>** ou **<input />** de type submit). Par exemple, pour demander à l'utilisateur de confirmer sa soumission, on peut procéder ainsi (ce code doit être placé après la déclaration de jQuery dans la page web) :

```
<script type='text/javascript'>
  // On attend que le DOM soit prêt
  $(document).ready(function(){
    // Dès que le document est soumis...
    $('#formulaire_test').submit(function(){
      // On affiche une fenêtre de confirmation
      var c = confirm("Confirmez-vous l'envoi des informations ?");
      // On retourne la réponse de l'utilisateur
      return c;
    });
  });
</script>
```

$('#formulaire_test').submit est déclenchée lorsque le formulaire portant l'identifiant « formulaire_test » (#formulaire_test) est soumis par l'utilisateur. confirm est une fonction qui affiche une boite dialogue de confirmation. Cette fonction confirm retourne les valeurs false ou true en fonction du choix de l'utilisateur, ce qui valide ou invalide le formulaire.

Voir aussi : https://api.jquery.com/submit/

IV.1.2. Change

La méthode change de jQuery permet de réagir lorsque la valeur d'un **<input />**, **<select></select>** ou le contenu d'un **<textarea></textarea>** ont été changés par l'utilisateur. Voici un exemple d'ouverture d'une boite d'alerte lorsque l'utilisateur fait un choix de radio bouton (ce code est à placer dans un $(document).ready(**function**(){...});) :

```
$("input[type='radio'][name='genre']").change(function(){
  alert("Vous avez fait un choix... Mais vous pourrez le modifier.");
});
```

$("input[type='radio'][name='genre']").change est déclenchée lorsque les **<input />** d'attribut « type » de valeur « radio » ([type='radio'] on utilise ici un nouveau sélecteur CSS avec des crochets qui permet de choisir un attribut particulier) et d'attribut « name » de valeur « genre » ([name='genre']) sont sélectionnés. Cela provoque l'ouverture d'une boite d'alerte grâce à la fonction alert.

Voir aussi : https://api.jquery.com/change/

IV.1.3. Focus, blur et keyup

Les deux méthodes focus et blur fonctionnent comme change, mais permettent de réagir lorsqu'un champ récupère (focus) ou perd (blur) le « focus ». Un champ (et plus généralement un élément HTML) a le « focus » quand l'utilisateur peut agir sur lui avec le clavier (si c'est un champ, on y voit le curseur texte).

La méthode keyup permet, de la même façon, de réagir à la fin de l'appui sur une touche (par exemple pour taper un caractère) dans un champ.

Ces méthodes s'utilisent comme change. Voir aussi :

- https://api.jquery.com/focus/
- https://api.jquery.com/blur/
- https://api.jquery.com/keyup/

IV.2. Récupérer les informations contenues dans le formulaire

Pour pouvoir vérifier le contenu des informations fournies par l'utilisateur ou réagir à ces informations, jQuery propose une méthode val pour lire le contenu du formulaire. En reprenant l'exemple de la méthode change, voici une utilisation de val :

```
$("input[type='radio'][name='genre']").change(function(){
  var choix_genre = $(this).val();
  alert("Vous avez fait le choix '"+choix_genre+"'... Mais vous pourrez le modifier.");
});
```

$(**this**).val() retourne le contenu de l'attribut value de l'élément courant (**this**, qui pointe sur l'élément qui déclenche l'événement). alert indique à l'utilisateur le choix fait grâce à la variable choix_genre. Il faut noter qu'en réutilisant le code présenté au début de ce document l'attribut value des radios boutons peut prendre les valeurs genre_femme, genre_homme, etc. Pour construire un message plus lisible pour les utilisateurs il faudra manipuler cette valeur avec, par exemple, un switch/case :

```
$("input[type='radio'][name='genre']").change(function(){
        var choix_genre = "";
        switch ($(this).val()) {
          case "genre_femme":
            choix_genre = "Femme";
            break;
          case "genre_homme":
            choix_genre = "Homme";
            break;
          case "genre_mauvais":
            choix_genre = "Mauvais";
            break;
          case "genre_autre":
            choix_genre = "Autre";
            break;
        }
        alert("Vous avez fait le choix '"+choix_genre+"'... Mais vous pourrez le modifier.");
      });
```

IV.3. Faire un retour (rapide) à l'utilisateur sur le contenu du formulaire

Pour finir, jQuery permet de modifier le contenu du formulaire en fonction des contenus récupérés par la méthode val. La manipulation des contenus du formulaire se fait comme celle du HTML en général (voir poly « *Initiation au JavaScript* »).

Examinez ce morceau de code qui s'applique au formulaire de date présenté dans le III.3 p.8 et essayez de comprendre comment il fonctionne.

```
      $('input:text#choix_date').keyup(function(){
        var expression_reguliere_date = '(0[1-9]|1[0-9]|2[0-9]|3[01])[- /.](0[1-9]|1[012])[- /.][0-9]{4}';
        var valeur_champ = $(this).val();
        // si la valeur du champ n'est pas une date...
        if (!valeur_champ.match(expression_reguliere_date)) {
          // On vérifie qu'on n'est pas déjà en train de s'en plaindre
          if (!$(this).parent('.form-group').hasClass('has-error')) {
            // Dans ce cas on retire un éventuel message de réussite
            $(this).siblings("strong.bon").remove();
            // et on affiche un message
            $(this).after("<strong class='pas_bon'>Cette date n'est pas (encore) correcte</strong>");
          }
          // dans tous les cas, on ajoute la classe d'erreur de Bootstrap
          $(this).parent('.form-group').addClass('has-error');
          // On retire la classe de réussite Bootstrap
          $(this).parent('.form-group').removeClass('has-success');
          // On retire aussi le message de succès s'il y en avait un
          $(this).siblings("strong.bon").remove();
        } else {
          // Si au contraire la date est bonne,
          // On regarde si on l'a déjà dit...
          if (!$(this).parent('.form-group').hasClass('has-success')) {
            // si non, on le dit...
            $(this).after("<strong class='bon'>date correcte</strong>");
          }
          // dans tous les cas on enlève la classe d'erreur de Bootstrap
          // et on ajoute celle de réussite
          $(this).parent('.form-group').removeClass('has-error');
          $(this).parent('.form-group').addClass('has-success');
          // On enlève le message d'erreur aussi
          $(this).siblings("strong.pas_bon").remove();
        }
      });
```

IV.4. Exercice 19

Modifiez le formulaire des exercices 17 et 18 pour proposer en JavaScript des interactions et une validation plus poussée des champs que l'auto-validation :

- Le texte du message doit faire entre 5 et 1000 caractères et l'utilisateur doit à chaque frappe savoir si son texte fait la bonne taille.
- Lorsque l'utilisateur choisit le thème « Autre thème » dans le menu déroulant, un champ lui permettant d'écrire un nouveau thème (libre) doit lui être proposé
- Une fois le formulaire validé, un boite de dialogue reprenant les données doit permettre à l'utilisateur de confirmer ou d'annuler.

V. Des plugins jQuery pour vous aider avec les formulaires

jQuery propose de nombreux plugins pour faciliter la gestion des formulaires et améliorer les interactions avec les utilisateurs. En voici quelques-uns :

- Des <select></select> plus intelligents et agréables : https://harvesthq.github.io/chosen/
- La même chose, mais adaptée à Bootstrap: http://nicolasbize.com/magicsuggest/
- Une gestion automatisée de la hauteur des **<textarea></textarea>** : http://www.unwrongest.com/projects/elastic/
- Un plugin de comptage des caractères dans un **<textarea></textarea>** http://tpgblog.com/2010/03/23/noblecount-jquery-character-count-plugin/
- Un retour *visuel* sur la force des mots de passe saisis http://benjaminsterling.com/password-strength-indicator-and-generator/
- Un retour *textuel* sur la force des mots de passe saisis http://bassistance.de/jquery-plugins/jquery-plugin-password-validation/
- De jolis boutons on/off à la place des boutons radio et des checkboxes http://widowmaker.kiev.ua/checkbox/
- Un plugin pour aider à la validation des formulaires et compatible Bootstrap http://bootstrapvalidator.com/
- Un plugin malin pour faciliter le choix d'une heure https://weareoutman.github.io/clockpicker/ (rappel: le plugin jQuery UI déjà proposé dans le poly « *Initiation au JavaScript* » propose aussi des outils pour les formulaires comme http://jqueryui.com/datepicker/ qui facilite la saisie des dates)
- Une interface sophistiquée pour permettre la saisie de durées http://www.dangrossman.info/2012/08/20/a-date-range-picker-for-twitter-bootstrap/
- Gérer des barres de progression http://www.minddust.com/project/bootstrap-progressbar/
- Permets de mettre en place une interface de tri d'objets https://johnny.github.io/jquery-sortable/
- Propose des outils pour « noter » des contenus http://plugins.krajee.com/star-rating
- Un sélecteur de couleurs malin (et adapté à Bootstrap) http://www.virtuosoft.eu/code/bootstrap-colorpickersliders/
- Une interface pour gérer des étiquettes (tags) http://welldonethings.com/tags/manager/v3