Classe «TelInput Mask »

Amélioration d'un input de type 'tel'

tel_input_mask.js

Version de la doc : 1.0

PRÉSENTATION

Permet d'appliquer un masque de saisie en temps réel.

Vérifie à chaque touche pressée si le numéro de téléphone saisi est valide ou non.

Possibilité d'accepter l'indicatif international ('+33'/'0033' pour la France et de le remplacer en temps réel par '0').

Possibilité d'accepter uniquement tel ou tel chiffre juste après (ex : '4' pour la région sud-est, '6' et '7' pour les mobiles).

ÉVÉNEMENTS

Trois événements sont déclenchés à chaque changement de validité (calculée dès qu'une touche est relachée) :

- '**change_validity**' : à chaque fois qu'un changement de validité a lieu
- '**valid**' : idem que 'change_validity' mais uniquement quand un numéro de téléphone **valide** est saisi
- '**invalid**' : idem que 'change_validity' mais uniquement quand un numéro de téléphone **invalide** est saisi

CLASSES

Deux classes sont appliquées à chaque changement de validité (calculée dès qu'une touche est relachée) :

- '**valid**' : quand un numéro de téléphone **valide** est saisi
- *'**invalid**' : quand un numéro de téléphone **invalide** est saisi*

CONFIG

De nombreux paramétres peuvent être personnalisés :

• '**franceOnly' {bool}** : si true, accepte uniquement les numéros à 10 chiffres commençant par '0', ou ceux avec l'indicatif '+33' ou '0033'.

• '**format' {bool}** : si true, formate le numéro lors de la saisie (ajoute un espace tous les deux chiffres sauf pour l'indicatif international s'il est saisi.

• '**delimiter' {string}** : il s'agit du texte qui sera inséré si l'option 'format' est activée. Un espace par défaut (' ').

• '**replaceInternationalIndicative' {bool}** : si true, l'indicatif international sera replacé par '0' (en instantané lors de la saisie).

• '**applyDynamicPattern' {bool}** : si true, un pattern est appliqué au champ dès qu'un changement de validité a lieu (vérifié lors de la saisie) :
- si numéro valide : un pattern identique à la valeur du champ est appliqué. Cela permet de faire en sorte que le navigateur interprète le champ comme valide.
- si numéro invalide : un pattern différent de la valeur du champ est appliqué. Cela permet de faire en sorte que le navigateur interprète le champ comme invalide (bordure rouge sans avoir eu besoin d'ajouter du css..)

• '**triggerEventsDuringInstantiation' {bool}** : si true, l'événement 'change_validity', ainsi que l'événement 'valid' ou 'invalid' sera déclenché dès l'appel au constructeur ou à la méthode statique add().

• '**pattern_secondDigit' {string}** : il s'agit du pattern qui concerne uniquement le second chiffre (le chiffre situé après le '0', ou après '+33' ou '0033' pour la France).
Par défaut il est défini à '4|6|7' pour accepter les numéros de la région sud-est et les numéros de téléphone mobile.

• ‘**pattern_charactersToFilter’ {string}** : il s’agit du pattern qui est appliqué à chaque caractère du champ dès qu’une touche est relachée, dans le but de filtrer les caractères à insérer dans le champ.
Par défaut il est défini à ‘[0-9+]’ pour filtrer les chiffres et ‘+’.

• ‘**pattern_internationalIndicativeCode’ {string}** : il s’agit du pattern qui correspond aux codes indicatifs internationaux qui sont acceptés.
Par défaut ce pattern est défini à ‘auto’, ce qui va lui permettre d’être auto-généré selon que l’option ‘franceOnly’ est activée ou non :
- si franceOnly activé : ‘33’
- si franceOnly désactivé : ‘[1-9]\\d’ (pour accepter tous les codes à deux chiffres qui commencent par un ‘1’)

• ‘**pattern_tel’ {string}** : il s’agit du pattern qui permet de vérifier si le numéro de téléphone saisi est valide ou non.
Par défaut ce pattern est défini à ‘auto’, ce qui va lui permettre d’être auto-généré comme ci-après :
- ‘0’ ou (‘+’ ou ‘00’ suivi de pattern_internationalIndicativeCode)
- suivi de pattern_secondDigit
- suivi de 8 chiffres

, à partir de, suivi de
- si franceOnly activé : ‘33’
- si franceOnly désactivé : ‘[1-9]\\d’ (pour accepter tous les codes à deux chiffres qui commencent par un ‘1’)

DÉPENDANCES

La présente classe nécessite d’avoir dans sa page (dans l’ordre suivant) :
- la librairie jQuery
- ‘utils.js’
- ‘tel_input_mask.js’

EXEMPLE DE MISE EN OEUVRE

Code html d'une page qui contient un input de type 'tel', et une div qui indique si le numéro est valide ou non, et une seconde qui affiche le numéro de tel extrait :

```
<!-- code html -->
<p>Accepte les numéros de tel commençant par '04', '06', '07'
, '0033'.. ou '+33'.. :</p>
<!-- le champ 'tel' -->
<input id="tel_input" type="tel">
<!--affichage des résultats :
numéro valide ou non, numéro de tel récupéré -->
<div id='status'>résultat</div>
<div id='valeur'>valeur</div>
```

Import des scripts JS nécessaires :

```
<!-- scripts js -->
<script src="http://code.jquery.com/jquery-
latest.min.js"></script>
<script src="utils.js"></script>
<script src="tel_input_mask.js"></script>
```

Codage du script JS de la page :

```
// au chargement de la page :
$(function() {
    // applique les nouveaux comportements au champ 'tel'
    let instance = TelInputMask.add('#tel_input');
    // dès qu'un changement de validité a lieu pendant la sai
sie
    $('#tel_input').on('change_validity', function() {
        // récupère la validité
        let isValid = $(this).is(':valid');
        let validityDescription = isValid ? 'Numéro VALIDE ☻'
 : 'mauvais numéro';
        // récupère la valeur sans espaces
        let value = instance.val();
```

```
        // affiche si le numéro est valide ou non, et le numé
ro récupéré
        $('#status').html(validityDescription );
        $('#valeur').html(value);
    })
})
```

>> En résumé :

La ligne suivante a suffit à appliquer les nouveaux comportements au champ, en prenant en compte la config par défaut :

```
TelInputMask.add('#tel_input');
```

On aurait pu appliquer une config différente comme ceci :

```
// on modifie la config pour :
//    - accepter uniquement les numéros de téléphonie fixes
//    - séparer les groupes par un point
let config = {
    'pattern_secondDigit':    '[1-5]',
    'delimiter':              '.'
};
// applique les nouveaux comportements au champ 'tel'
TelInputMask.add('#tel_input', config);
```

MÉTHODES STATIQUES

• add (input, config = ‘auto’)

Crée une instance, applique le comportement attendu au champ input, à partir de la config fournie.

• Le champ *input* peut être :

- un sélecteur CSS. Ex :
  ‘#telInput’

- un élément du DOM. Ex :
  document.getElementById(‘telInput’)

- un objet jQuery. Ex :
  $(‘#telInput’)

• Vous pouvez transmettre une *config* comprenant vos paramètres personnalisés, config à transmettre sous forme d’objet JSON :

- Si ‘auto’, tous les paramètres par défaut seront appliqués.

- Si JSON, la config par défaut sera appliquée mais vos paramètres personnalisés écraseront les valeurs par défaut.

• obj (input)

Retourne l’instance associée à un input. Ex :
TelInputMask.obj(‘#telInput’) // retournera l’instance liée au champ ‘#telInput’

• getDefaultConfig ()

Retourne la config par défaut. Utile si vous souhaitez modifier certains paramètres avant d’appeler le constructeur.

• setDefaultConfig (config)

Modifie la config par défaut. Utile si vous avez de nombreux champs avec une config commune différente de celle par défaut.

• start ()

Reprends l'application des comportements sur tous les champs passés au constructeur. Utile seulement si un *stop()* a été effectué.

• stop ()

Stoppe l'application des comportements sur tous les champs passés au constructeur.

MÉTHODES D'INSTANCE

Ces méthodes sont liées à un seul champ input.

• getConfig ()

Retourne la config liée au champ.

• start ()

Reprends l'application des comportements sur le champ. Utile seulement si un *stop()* a été effectué.

• stop ()

Stoppe l'application des comportements sur le champ.

• val ()

Si le numéro de téléphone saisi par l'utilisateur est valide, celui-ci sera retourné sans les espaces ni caractères superflus.

PRINCIPALES POSSIBILITÉS D'AMÉLIORATION

• Gestion plus performante du 'annuler / rétablir' (ctrl + z / ctrl + y)

Actuellement cet outil est capable de rétablir la dernière valeur lorsque l'utilisateur appui sur 'ctrl + z', mais cela se limite à une valeur.

Aussi, la gestion du rétablir (ctrl+y) n'est pas assurée.

• Faire en sorte que les caractères non retenus par le filtre n'apparaissent pas du tout

Actuellement les caractères saisis autres que des chiffres apparaissent une fraction de seconde. Cela est dû au fait que le filtre agit lors du relâchement des touches. Traiter cela lorsque la touche est pressée aurait permis d'éviter cela mais c'est plus lourd à mettre en œuvre (vérifier les codes ascii, autres touches pressées simultanément…).

• Mise en place d'un attribut maxlength en option

UTILISATION

J'espère que cet outil vous plaira, je vous en souhaite une bonne utilisation 😊

Made in France
with ♥

#happy_coders