# Champs de saisie

*Note : Il s'agit des champs de type* `TEXTAREA` *et* `INPUT[type="text"]` *.*

Les méthodes de l'objet `UI.Input` permettent de traiter toutes les éditions quel que soit le champ de saisie, qu'il appartienne ou non à une fiche.

## Observer un champ d'édition

### Champ unique

Pour un champ unique, il suffit de faire :

```
UI.Input.bind(<DOMElement | jQuerySet>)
```

### Champs d'un container

Même code pour un container (i.e. un panneau contenant des champs de saisie textuels).

```
UI.Input.bind(<DOMElement | jQuerySet du container>)
```

## Traitement on-the-fly de la donnée

En définissant dans les attributs du champ les attributs `data-type` et `data-format` , on peut définir :

- La validité d'une donnée ;
- La transformation à la volée de la valeur d'une donnée.

### Data types :

```
number      La donnée doit être un nombre pour être valide
horloge     La donnée sera transformée en horloge H:MM:SS
people      La donnée est une liste de personnes (nécessite
            un data-format)
```

Exemple :

```
<input
    type="text"
    id="un_textfield"
    data-type="number"
    data-format="[0-9]{2}"
    />
```

*Note : Si de nouveaux types doivent être créés, les ajouter dans* `UI.Input.check_value` *.*

## Data format

L'attribut `data-format` dans le code HTML du champ de saisie permet de définir le format de la donnée.

Dans le cas généraux (hors data-type comme `people` définissant un data-format par identifiant), `data-format` est une expression régulière qui sera comparée à la valeur du champ par l'opération :

```
OK = valeur_champ.replace(<regexp data-format>) == ""
```

Par exemple, pour définir un champ qui ne peut contenir qu'une année, sur 4 lettres, commençant obligatoirement par 19 ou 20 (pour des années du XX ou XXIe siècle) :

```
<input
    type="text"
    ...
    data-type="number"
    data-format="(19|20)[0-9]{2}"
    />
```

### Data format pour le data-type `people`

Pour une donnée de type `people`, le `data-format` peut être :

```
undefini    prénom, nom
auteur      prénom, nom, objet (scénario, roman, etc.)
acteur      prénom, nom, prénom/surnom personnage, nom
            personnage, fct personnage
```

## Champs de saisie des fiches

`UI.Input` reconnait un champ de saisie appartenant à une fiche grâce à l'identifiant de ce champ qui commence toujours par `f-<id fiche>`. De la même manière, la 3e donnée de l'identifiant doit définir la propriété de la fiche modifiée :

```
id_champ = f-<id fiche>-<propriété fiche>
```

Cela permet à `UI.Input.check_value` de faire des traitements de détail, ou à la méthode `UI.Input.keypress` d'agir différemment suivant la propriété.

### Méthode onchange

Au blur du champ de saisie, s'il appartient à une fiche, `UI.Input` va rechercher une méthode `onchange_<property>`. S'il la trouve, elle sera invoquée pour prendre en compte le changement de valeur (if any).

Par exemple, dans une fiche de type livre qui aurait un identifiant #12, le champ de titre réel est défini par :

```
<input
  type="text"
  id="f-12-real_titre"
  value=""
  />
```

Ce champ doit être suivi par `UI.Input` grâce à :

```
UI.Input.bind($('f-12-real_titre'))
```

`UI.Input` analyse la cible ( `UI.Input.target` ) et définit :

```
target.hasFiche = true
target.fiche_id = 12
target.property = 'real_titre'
```

Ensuite, lorsqu'on blur de ce champ de saisie, `UI.Input` va chercher la méthode :

```
Book.prototype.onchange_real_titre
```

Il va la trouver, et lui envoyer la nouvelle valeur du titre réel.