Phalcon-jquery : JQuery

Principales fonctionnalités de la librairie Phalcon-JQuery (hors JQuery UI et Twitter Bootstrap)

-- Association d'évènements

-- Evénements connus

-- Exemple

Click sur un bouton d'id **#btn** :

```
    public function clickAction(){
        $this->click("#btn","console.log('click sur #btn');");
        $this->query->compile($this->view);
    }
```

Vue associée :

```
<input type="button" value="Test click" id="btn">
{{script_foot}}
```

-- Utilisation du paramètre event du callback

Le code passé au callback peut utiliser le paramètre **event** passé par JQuery, et **$(this)**, la cible de l'évènement (target) :

Exemple : appui sur une touche, récupération du code de la touche frappée et de l'id de l'élément ayant généré l'évènement.

```
    public function clickAction(){
        $this->jquery->keydown("#btn","console.log('code de touche appuyée :' +
event.which + ' sur #' + $(this).attr('id'));");
        $this->jquery->compile($this->view);
    }
```

Exemple : affichage/masquage d'un élément sur changement de la valeur d'une checkbox :

```
    public function checkedAction(){
$this->jquery->change("#ck",$this->jquery->toggle("#fs","$(this).checked()"));
        $this->jquery->compile($this->view);
    }
```

Vue associée :

```
<input type="checkbox" id="ck" checked>
<fieldset id="fs">
<legend>Elément à afficher/masquer</legend>
</fieldset>
{{script_foot}}
```

-- Liste des méthodes JQuery liées aux évènements

Méthode	**Evènement**
blur(selector,callback)	Sur perte du focus par l'élément désigné par **selector**
change(selector,callback)	Sur changement de valeur d'un **input**, **select** ou **textarea** (déclenché généralement sur perte focus)
click(selector,callback)	Sur click de l'élément désigné par **selector**
dblclick(selector,callback)	Sur double-click de l'élément désigné par **selector**
error(selector,callback)	Sur erreur de chargement sur **selector**
focus(selector,callback)	Sur réception du focus par l'élément désigné par **selector**
hover(selector,callback)	Sur déplacement de la souris au dessus de l'élément désigné par **selector**
keydown(selector,callback)	Sur touche clavier enfoncée lorsque l'élément désigné par **selector** a le focus
keypress(selector,callback)	Sur touche clavier enfoncée lorsque l'élément désigné par **selector** a le focus
keyup(selector,callback)	Sur touche clavier relachée lorsque l'élément désigné par **selector** a le focus
load(selector,callback)	Sur chargement de l'élément désigné par **selector**
mousedown(selector,callback)	Sur bouton souris enfoncé lorsque l'élément désigné par **selector**
mouseout(selector,callback)	Sur sortie de la souris de l‘‘élément désigné par **selector**
mouseover(selector,callback)	Sur passage souris au dessus de l'élément désigné par **selector**
mouseup(selector,callback)	Sur bouton souris relaché lorsque l'élément désigné par **selector**
ready(callback)	Sur chargement du document terminé
scroll(selector,callback)	Sur défilement dans/sur l'élément désigné par **selector**
trigger(selector,event)	Déclenche l'évènement **event** sur l'élément désigné par **selector**
unload(selector,callback)	Sur déchargement de l'élément désigné par **selector**

-- Evènements non listés

Pour les évènements inconnus de JQuery ou ne disposant pas de méthode associée dans la librairie phalcon-jquery, on utilise les méthodes :

- execAndBindTo(selector,event,callback) permettant d'exécuter le **callback** sur le déclenchement de **event** sur l'élément désigné par **selector**.
- doJqueryAndBindTo($element,$event,$elementToModify,$jqueryCall,$param="",$function="") permettant d'exécuter la méthode jquery **jqueryCall** sur **elementToModify**, en lui passant **param** sur le déclenchement de **event** sur l'élément désigné par **element**.

-- Requêtes Ajax

-- Requêtes immédiates

Il s'agit des requêtes dont l'exécution n'est pas différée (non associée à un évènement par exemple).

Les requêtes immédiates peuvent permettre le chargement d'une page en cascade : élément par élément.

--GET

Exemple : Chargement de **exemple/getResponse** dans la div **#divResponse**

```
    public function getAction(){
        $this->jquery->get("exemple/getResponse","#divResponse");
        echo "<div id='divResponse'>Chargement...</div>";
        echo $this->jquery->compile();
    }
    public function getResponseAction(){
        echo "Chargement terminé";
        $this->view->disable();
    }
```

-- POST

Exemple : Post de valeurs vers **exemple/postResponse** dans la div **#divPostResponse**

```
    public function postAction(){
$this->jquery->post("testsbs/postResponse","#divPostResponse","{nom:'DOE',prenom:'J
ohn'}");
        echo "<div id='divPostResponse'>Chargement...</div>";
        echo $this->jquery->compile();
    }
    public function postResponseAction(){
        var_dump($_POST);
        $this->view->disable();
    }
```

-- JSON

Les requêtes JSON permettent d'alimenter en données une vue, à partir d'un retour d'objet JSON par le serveur.

Exemple : Le get JSON vers la page **exemple/jsonResponse** alimente tous les éléments DOM **nom** et **prenom** avec l'élément JSON retourné.

```
    public function jsonAction(){
        $this->jquery->json("testsbs/jsonResponse");
        echo "<div id='nom'>Chargement...</div>";
        echo "<input type='text' id='prenom' value='chargement...'>";
        echo $this->jquery->compile();
    }
    public function jsonResponseAction(){
        echo '{"nom":"DOE","prenom":"John"}';
        $this->view->disable();
```

```
    }
```

-- Association de requêtes à des évènements

Pour faire plus court, les exemples suivants ne comportent pas de vues, et les éléments HTML sont directement produits dans le contrôleur (à éviter...)

-- getAndBindTo

Permet d'associer l'exécution d'un Get à un évènement déclenché sur un élément :

Exemple : Sur click du bouton **#btnGet**, on veut effectuer un get vers **exemple/getResponse**, et afficher le résultat dans la div **#divResponse**.

```
    public function getAction(){
$this->jquery->getAndBindTo("#btn","click","testsbs/getResponse","#divResponse");
        echo "<input type='button' id='btn' value='doGet'>";
        echo "<div id='divResponse'>En attente de click...</div>";
        echo $this->jquery->compile();
    }
    public function getResponseAction(){
        echo "Chargement terminé";
        $this->view->disable();
    }
```

-- postFormAndBindTo

Permet d'associer le post d'un formulaire à un évènement déclenché sur un élément :

Exemple : Sur click du bouton **#btnPost**, on veut poster le formulaire **frmLogin** vers **exemple/postResponse**, et afficher le résultat dans la div **#divResponse**.

```
    public function postResponseAction(){
        var_dump($_POST);
        $this->view->disable();
    }

    public function postFormAction(){
$this->jquery->postFormAndBindTo("#btn","click","testsbs/postResponse","frmLogin","
#divPostResponse");
        echo "<form id='frmLogin' name='frmLogin'  onsubmit='return false;'><input
type='text' name='login' placeholder='Login...'>
            <input type='password' name='password'
placeholder='password...'></form>";
        echo "<input type='button' id='btnPost' value='doPost'>";
        echo "<div id='divPostResponse'>Avant validation</div>";
        echo $this->jquery->compile();
    }
```

Il est également possible d'utiliser :

Méthode	Rôle
postAndBindTo(element,event,url[,params,responseElement])	Post les paramètres **params** vers url sur l'**event** produit sur **element**

Ou de combiner les méthodes différées avec les méthodes évènementielles :

Requête Ajax Get sur click d'un bouton :

```
    public function getAction(){
$this->jquery->click("#btn",$this->jquery->getDeferred("testsbs/getResponse","#divR
esponse"));
        echo "<input type='button' id='btn' value='doGet'>";
        echo "<div id='divResponse'>En attente de click...</div>";
        echo $this->jquery->compile();
    }
```

-- Manipulation du DOM

Méthode	Rôle
append(to,element)	ajoute l'élément DOM **element** à la fin de chaque élément désigné par **to**
prepend(to,element)	ajoute l'élément DOM **element** au début de chaque élément désigné par **to**
attr(selector,attributeName [,value,immediatly])	Lit/modifie l'attribut **attributeName** des éléments correspondant à **selector**
html(selector[,value,immediatly])	Lit le contenu html / Affecte **value** au contenu html des éléments correspondant à **selector**

-- Utilitaires

Méthode	Rôle
show(selector[,speed,callback])	Affiche le ou les éléments DOM correspondant au selecteur **selector**
hide(selector[,speed,callback])	Masque le ou les éléments DOM correspondant au selecteur **selector**
fadeIn(selector[,speed,callback])	Effectue une transition sur le ou les éléments DOM correspondant au selecteur **selector** pour le(s) rendre opaque
fadeOut(selector[,speed,callback])	Effectue une transition sur le ou les éléments DOM correspondant au selecteur **selector** pour le(s) rendre transparent
toggle(selector)	Affiche/masque le ou les éléments DOM correspondant au selecteur **selector**
animate(selector[,params,speed,extra])	Effectue une animation sur le ou les éléments DOM correspondant au selecteur **selector**
addClass(selector,class[,immediatly])	Ajoute la classe **class** sur les éléments DOM correspondant au selecteur **selector**
toggleClass(selector,class[,immediatly])	Ajoute/supprime la classe **class** sur le ou les éléments DOM correspondant au selecteur **selector**