Libs Docs

Bastien Helec

09/07/2025

JS méthodes

```
Bastien Helec
Description : Explication de la méthode css

08-07-2025
```

Sommaire

Model

Description:

Le nom des fichiers correspondent au classe comme :

- remove_event.php ⇒ JS_Remove_Event
- onclick_event.php ⇒ JS_Onclick_Event
- calendar_get.php ⇒ JS_GET_CALENDER

Classes

JS_Remove_Event :

Description:

Cette classe permet de supprimer de manière générale un événement “actif” *exemple: un formulaire pop-up*

Arguments du constructeur :

`$div_id` : Chaîne de caractères (String) représentant l'identifiant (Généralement des div) cible a retirer de l'affichage.

`$btn_id` : Chaîne de caractères (String) représentant l'identifiant du btn qui permet d'appeler le contenu de `$div_id`

Exemple d'utilisation :

```php
<?php
$Connexion_Utilisateur = new JS_Remove_Event('Connexion_div', 'Connexion_btn');
echo $Connexion_Utilisateur->gen_remove_js();
?>
```

Cela produira :

```javascript
document.addEventListener('click', (event) => {
    if (!Connexion_div.contains(event.target) && Connexion_btn !==event.target ){
        Connexion_div.classList.remove('deplacer');
        setTimeout(()) => {
            Connexion_div.classList.remove('actif');
            }, 100);
        }
    });
```

Méthodes principales

`gen_remove_js` : génère le code javascript permettant l'action de la classe.

JS_Onclick_Event :

Description :

Cette classe permet d'effectuer une requêtes POST pour afficher une données provenant de la zone sélectionner si il contient un attribut data-*.

- Génère un script JavaScript dynamique.
- Gère les clics sur des boutons par classe ou ID.
- Récupère un identifiant via un attribut data-*.
- Envoie une requête AJAX POST vers un fichier PHP.
- Affiche ou masque un formulaire cible.
- Gère l'état d'une bannière associée.

exemple:

Un bouton contient des informations de reservations de salle ainsi qu'un attribut dans data-.*

- La classe va alors récupérer la valeur de data- et l'envoyer dans le fichier que gère le back-end et le traitement de la données.*

- Le back va ensuite renvoyer des informations qui va générer la fenêtre.

Arguments du constructeur :

- `$btn_id_class` : Chaîne de caractères (String) représentant le sélecteur CSS (classe ou ID) des boutons à écouter pour le clic.
- `$file` : Chemin du fichier PHP cible pour la requête AJAX.
- `$formulaire_to_display_id` : Identifiant (String) du conteneur de formulaire à afficher/masquer.
- `$banner` : Identifiant (String) de la bannière à manipuler lors de l'action.

Exemple d'utilisation :

```php
<?php
$event = new JS_Onclick_Event('.btn-demande', 'demande_autorisation.php', 'formulaire_autorisation', 'banne
echo $event->getID_from_sameclass_js('demande');
?>
```

Cela produira (extrait simplifié) :

```javascript
document.addEventListener('DOMContentLoaded', function() {
    const buttons = document.querySelectorAll('.btn-demande');
    const formContainer = document.getElementById('formulaire_autorisation');
    const banner = document.getElementById('banner_info');
    let currentId = null;

    if (!formContainer) {
        console.warn('Element cible non trouvé : #formulaire_autorisation');
        return;
    }

    buttons.forEach(btn => {
        btn.addEventListener('click', function(event) {
            event.stopPropagation();
            let id = this.dataset['demande'] || this.dataset.id;
            if (!id) {
                console.warn('ID non trouvé (ni data-demande, ni data-id)');
                return;
            }
            fetch('demande_autorisation.php', {
                method: 'POST',
                headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
                body: 'id=' + encodeURIComponent(id)
            })
            .then(response => response.text())
            .then(html => {
                if (id !== currentId && currentId !== null) {
                    location.reload();
                    return;
                } else {
                    if (formContainer.classList.contains('actif')) {
                        formContainer.style.display = 'none';
                        formContainer.classList.remove('actif', 'deplacer');
                        currentId = id;
                    } else {
                        formContainer.style.display = 'block';
                        formContainer.classList.add('actif');
                        setTimeout(() => {
                            formContainer.classList.add('deplacer');
                        }, 100);
                        currentId = id;
                    }
                }
                if (banner) {
                    banner.classList.remove('actif');
                }
            })
            .catch(error => console.error('Erreur lors de la requête :', error));
        });
    });
});
```

Méthodes principales

`getID_from_sameclass_js` : C'est une méthode unique faisant la gestion de toute la classe (pas de méthodes annexes dissociant le besoin).

JS_Get_Calendar :

Description:

Classe annexe permettant d'introduire un calendrier extérieur.

exemple: Un calendrier Nextcloud.

Arguments du constructeur :

- `$calendar_name_url` : Tableau associatif où chaque clé est le nom d'une salle (ou d'une ressource) et chaque valeur est l'URL du calendrier correspondant.
- `$calendar_id` : Identifiant (String) de la liste déroulante (select) permettant de choisir la salle.
- `$calender_iframe_id` : Identifiant (String) de l'iframe où le calendrier sera affiché.
- `$calender_start_url` : Chaîne de caractères représentant le préfixe d'URL à ajouter avant chaque URL de calendrier.

Exemple d'utilisation :

```php
<?php
$calendars = [
    'Salle A' => 'calendarA.ics',
    'Salle B' => 'calendarB.ics'
];
$calendar = new JS_GET_CALENDAR($calendars, 'select_salle', 'iframe_calendrier', '/calendriers/');
echo $calendar->getCalendarJS();
?>
```

Cela produira (extrait simplifié) :

```javascript
document.addEventListener('DOMContentLoaded', function() {
    const calendarMap = {"Salle A":"calendarA.ics","Salle B":"calendarB.ics"};
    const calendarSelect = document.getElementById('select_salle');
    const calendarIframe = document.getElementById('iframe_calendrier');
    const calendarStartUrl = '/calendriers/';
    if (calendarIframe == null){
        console.log('Pas de zone de calendrier à afficher');
    } else {
        calendarIframe.style.display = 'none';
        calendarSelect.addEventListener('change', function() {
            const selectedSalle = this.value;
            const calendarUrl = calendarMap[selectedSalle];
            if (calendarUrl) {
                calendarIframe.src = calendarStartUrl + calendarUrl;
                console.log('Calendrier affiché :', selectedSalle);
                calendarIframe.style.display = 'block';
            } else {
                calendarIframe.src = '';
                calendarIframe.style.display = 'none';
                console.warn('Aucun calendrier trouvé pour la salle sélectionnée.');
            }
        });
    }
});
```

Méthodes principales

`getCalendarJS` : Génère le code JavaScript permettant d'afficher dynamiquement un calendrier externe dans une iframe selon la sélection de l'utilisateur.

FORMULAIRES Le formulaires contient les points suivants:

- call_banner.php ⇒ JS_Call_Banner
- call.php ⇒ JS_CALL_FORM
- ListeBDD.php ⇒ JS_Sort_ListeBDD
- remove_banner.php ⇒ JS_Remove_Banner
- Send.php ⇒ Send

JS_Call_Banner

Description :
Cette classe permet d'afficher dynamiquement une bannière d'information ou de notification sur la page, en modifiant son contenu et en lui appliquant une classe CSS pour la rendre visible.

Arguments du constructeur :

- `$id_banner` : Chaîne de caractères représentant l'identifiant de la bannière à cibler (par exemple, `banner_info`).
- `$message` : Chaîne de caractères correspondant au message à afficher dans la bannière.

Exemple d'utilisation :

```php
<?php
$banner = new JS_Call_Banner('banner_info', 'Demande d\'autorisation envoyée avec succès !');
echo $banner->Call_Banner_js();
?>
```

Cela produira (extrait simplifié) :

```
banner_info.textContent = 'Demande d\'autorisation envoyée avec succès !';
banner_info.classList.add('actif');
```

Méthodes principales

`Call_Banner_js` : Génère le code JavaScript permettant de modifier le texte d'une bannière et de l'afficher en ajoutant la classe CSS

JS_CALL_FORM

Description :
Cette classe permet d'afficher dynamiquement un formulaire (ou une div) lors du clic sur un bouton spécifique, en appliquant les classes CSS nécessaires pour l'animation et en masquant la bannière d'information si elle est présente.

Arguments du constructeur :

- `$id_div` : Chaîne de caractères représentant l'identifiant de la div ou du formulaire à afficher.
- `$id_btn` : Chaîne de caractères représentant l'identifiant du bouton qui déclenche l'affichage.
- `$banner` : Chaîne de caractères représentant l'identifiant de la bannière à masquer lors de l'ouverture du formulaire.

Exemple d'utilisation :

```php
<?php
$form = new JS_CALL_FORM('formulaire_contact', 'btn_ouvrir_form', 'banner_info');
echo $form->gen_print_FORM_js();
?>
```

Cela produira (extrait simplifié) :

```
btn_ouvrir_form.addEventListener('click', (event) => {
    event.preventDefault();
    formulaire_contact.classList.add('actif');
    formulaire_contact.style.display = 'block';
    setTimeout(() => {
        formulaire_contact.classList.add('deplacer');
    }, 100);
    banner_info.classList.remove('actif');
});
```

Méthodes principales

`gen_print_FORM_js` : Génère le code JavaScript permettant d'afficher dynamiquement un formulaire et de masquer la bannière.

JS_Sort_ListeBDD

Description :
Cette classe permet de gérer dynamiquement l'affichage d'une liste déroulante personnalisée (type select) et la sélection d'une valeur dans une base de données ou une liste d'options.

Arguments du constructeur :

- `$id_div` : Chaîne de caractères représentant l'identifiant principal de la liste personnalisée.

Exemple d'utilisation :

```
<?php
$liste = new JS_Sort_ListeBDD('ma_liste');
echo $liste->gen_list_js();
?>
```

Cela produira (extrait simplifié) :

```
if (ma_liste_select && ma_liste_options && ma_liste_choix_btn && ma_liste_choix_valeur) {
    ma_liste_choix_btn.addEventListener('click', (event) => {
        event.preventDefault();
        ma_liste_select.classList.toggle('actif');
    });

    ma_liste_options.addEventListener('click', (event) => {
        event.preventDefault();
        if (event.target.tagName === 'LI') {
            ma_liste_choix_valeur.value = event.target.getAttribute('data-value');
            ma_liste_choix_btn.textContent = event.target.textContent;
            ma_liste_select.classList.remove('actif');
        }
    });
}
```

Méthodes principales

`gen_list_js` : Génère le code JavaScript pour gérer l'ouverture, la sélection et la fermeture d'une liste personnalisée.

JS_Remove_Banner

Description :
Cette classe permet de masquer dynamiquement une bannière d'information ou de notification lors d'un clic n'importe où sur la page.

Arguments du constructeur :

- `$id_banner` : Chaîne de caractères représentant l'identifiant de la bannière à masquer.

Exemple d'utilisation :

```php
<?php
$remove = new JS_Remove_Banner('banner_info');
echo $remove->Remove_Banner_js();
?>
```

Cela produira (extrait simplifié) :

```javascript
document.addEventListener('click', (event) => {
    banner_info.classList.remove('actif');
});
```

Méthodes principales

`Remove_Banner_js` : Génère le code JavaScript permettant de masquer la bannière lors d'un clic.

Send

Description :
Cette classe permet de gérer l'envoi d'un formulaire via AJAX, de traiter la réponse du serveur, d'afficher une bannière de notification, et éventuellement de rediriger l'utilisateur après succès.

Arguments du constructeur :

- `$id_form` : Chaîne de caractères représentant l'identifiant du formulaire à gérer.
- `$path` : Chemin du fichier PHP qui traite la soumission du formulaire.
- `$resultpath` : (Optionnel) Chemin de redirection après succès.

Exemple d'utilisation :

```php
<?php
$send = new Send('formulaire_contact', 'traitement.php', 'merci.html');
$send->Send();
?>
```

Cela produira (extrait simplifié) :

```javascript
document.addEventListener('DOMContentLoaded', function() {
    // Gestion du bouton submit et du formulaire
    // Envoi AJAX avec FormData
    // Affichage de la bannière selon la réponse
    // Redirection éventuelle après succès
});
```

Méthodes principales

`Send` : Génère et affiche le code JavaScript pour gérer l'envoi AJAX du formulaire, l'affichage de la bannière

---

Vue-Implémentation

Il n'y a pas de section vue afin de pouvoir l'utiliser il faut alors créer un Controller_JS contenant :

```php
header("Content-Type: text/javascript");
```

Ce paramètre permet d'indiquer au serveur PHP l'utilisation de ce controller comme un CSS.

Le serveur considérera ce fichier comme une feuille `script.js`.