SimpleHtmlEditor

Mode d’emploi de la librairie PHP/JS autonome

Éditeur riche léger · sans npm · sans framework · stockage local ou base MySQL

Objectif

Ce guide explique comment installer la librairie, créer un éditeur, récupérer le HTML formaté, puis sauvegarder les textes soit dans le navigateur, soit dans une base de données via un bouton de sauvegarde.

1. Installation rapide

Télécharge le fichier simple-html-editor.js depuis HTMLEditor, puis copie-le dans ton projet, par exemple dans assets/js/.

HTML minimal

```
<div id="editor-root"></div>
<script src="assets/js/simple-html-editor.js"></script>
```

2. Initialisation de l’éditeur

Créer une instance

```
var editor = new SimpleHtmlEditor(document.getElementById("editor-root"));
```

Tu peux aussi fournir un contenu HTML initial au chargement de la page.

Avec contenu initial

```
var editor = new SimpleHtmlEditor("#editor-root", {
  initialHtml: "<p>Texte de départ</p>"
});
```

3. Récupérer le contenu HTML

La méthode getHtml() renvoie le HTML formaté produit par l'éditeur. C'est cette valeur que tu peux stocker dans localStorage, envoyer à PHP, ou enregistrer en base de données.

Lecture du contenu

```
var html = editor.getHtml();
```

4. Sauvegarde locale dans le navigateur

Pour un prototype ou un test rapide, tu peux enregistrer chaque texte dans localStorage. Les données restent dans le navigateur de l'utilisateur.

Bouton de sauvegarde locale

```
<button id="save-local">Sauvegarder localement</button>
<script>
var editor = new SimpleHtmlEditor("#editor-root");
document.getElementById("save-local").onclick = function () {
  var html = editor.getHtml();
  window.localStorage.setItem("mon-texte-html", html);
};
</script>
```

Recharger le texte local

```
var html = window.localStorage.getItem("mon-texte-html") || "<p></p>";
editor.setHtml(html);
```

5. Sauvegarder en base avec un bouton

Pour une vraie application, le bouton de sauvegarde peut envoyer le titre et le HTML vers une route PHP. PHP reçoit les données, vérifie le contenu, puis insère ou met à jour la base MySQL.

Page HTML + JavaScript

```
<input id="text-title" type="text" placeholder="Titre">
<div id="editor-root"></div>
<button id="save-db">Sauvegarder en base</button>
<script src="simple-html-editor.js"></script>
<script>
var editor = new SimpleHtmlEditor("#editor-root");
document.getElementById("save-db").onclick = function () {
  fetch("save_text.php", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      title: document.getElementById("text-title").value,
      html: editor.getHtml()
    })
  })
  .then(function (response) { return response.json(); })
  .then(function (result) {
    if (!result.ok) { alert("Erreur de sauvegarde"); return; }
    alert("Texte sauvegardé");
  });
};
</script>
```

6. Route PHP save_text.php

Exemple côté serveur

```
<?php
header("Content-Type: application/json; charset=utf-8");
$payload = json_decode(file_get_contents("php://input"), true);
$title = trim($payload["title"] ?? "");
$html = $payload["html"] ?? "";

if ($title === "") {
    http_response_code(422);
    echo json_encode(["ok" => false, "error" => "Titre manquant"]);
    exit;
}

$pdo = new PDO("mysql:host=localhost;dbname=ma_base;charset=utf8mb4", "user",
"password");
$stmt = $pdo->prepare("INSERT INTO texts (title, html, updated_at) VALUES (?,
?, NOW())");
$stmt->execute([$title, $html]);

echo json_encode(["ok" => true, "id" => $pdo->lastInsertId()]);
```

7. Table MySQL minimale

Schéma de départ

```
CREATE TABLE texts (
  id INT AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  html MEDIUMTEXT NOT NULL,
  updated_at DATETIME NOT NULL
);
```

8. Modifier un texte existant

Quand tu ouvres un texte existant, récupère son HTML depuis la base puis recharge l'éditeur avec setHtml(). Au moment de sauvegarder, renvoie aussi l'identifiant du texte.

Côté navigateur

```
editor.setHtml(htmlDepuisLaBase);

fetch("save_text.php", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ id: textId, title: title, html: editor.getHtml() })
});
```

Côté PHP : UPDATE

```
UPDATE texts
SET title = ?, html = ?, updated_at = NOW()
WHERE id = ?
```

Sécurité importante

Le contenu HTML vient du navigateur. Pour une application ouverte à plusieurs utilisateurs, nettoie le HTML côté serveur ou limite les balises autorisées afin d'éviter l'injection de JavaScript.

9. Fonctions disponibles

gras, italique, souligné

listes à puces et listes numérotées

- titres 1 à 6 et paragraphe
- polices courantes PC/Mac
- alignements gauche, centre, droite
- liens, couleur du texte et nettoyage de format

10. API JavaScript

- editor.getHtml() : récupère le HTML actuel
- editor.setHtml(html) : remplace le contenu de l’éditeur
- editor.clear() : vide l’éditeur

À retenir

La librairie est autonome : pas de npm, pas de framework, pas de jQuery. Elle utilise contenteditable et document.execCommand pour rester simple et compatible avec un hébergement PHP/Plesk classique.