Exploration

Les éditeurs de texte riche

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision

Sommaire

Nomenclature :

- **Assimiler** : cours pur. Explication théorique et détaillée (globalement supérieur à 4 pages).
- **Décoder** : fiche de cours, généralement inférieure à 5 pages.
- **Découvrir** : Travaux dirigés. Faisable sans matériel.
- **Explorer** : Travaux pratiques. Nécessite du matériel ou des logiciels.
- **Mission** : Projet encadré ou partie d'un projet.
- **Voyager** : Projet en autonomie totale. Environnement ouvert : Vous êtes le capitaine !

A Introduction

La saisie d'information dans un site web est souvent plus pratique que les formulaires de base, présents en HTML 5.

Nous sommes habitués aux éditeurs de texte complets, permettant de faire de la mise en forme et de rendre nos écrits plus agréables : gras, italique, souligné, mais aussi liste à puces et insertion d'images.

A.1 Présentation

Il existe bien évidemment des frameworks capables de proposer de tels éditeurs de texte, comme un simple formulaire et s'intégrant à nos pages web.

Dans cette activité simple, nous allons en utiliser trois différents et votre travail sera de les comparer pour déterminer les avantages et inconvénients de chacun.

A.2 Prérequis

Aucun

B Présentation des trois frameworks

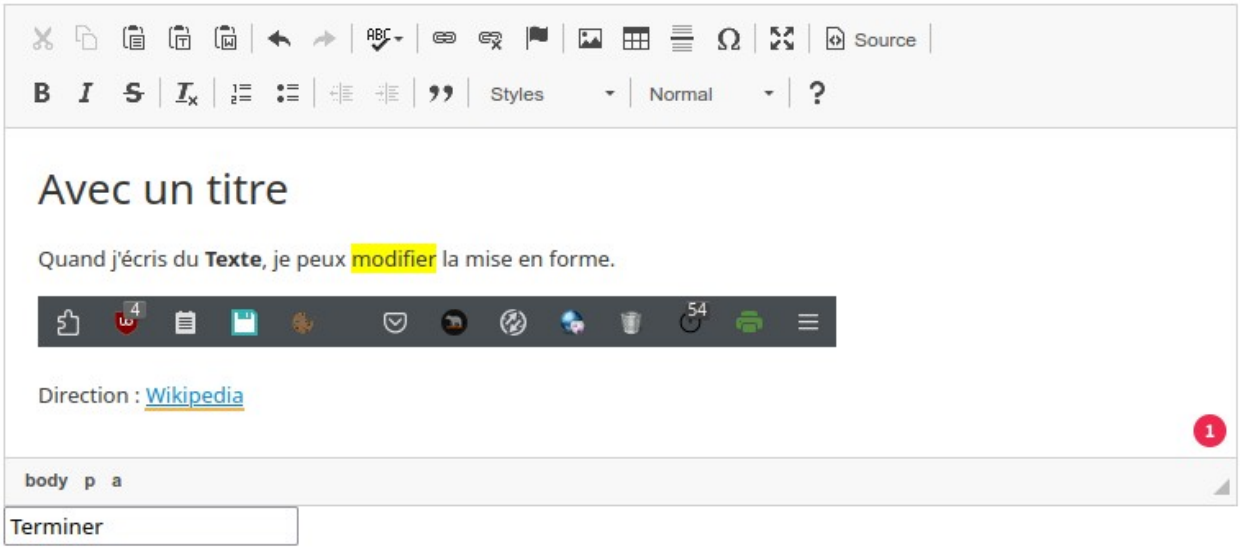
Les trois éditeurs qui sont étudiés ici, sont tous WYSIWYG (What You See Is What You Get).

B.1 CKEditor

B.1.1 Généralités

L'éditeur de https://ckeditor.com/ est livré sous la licence GPL2+ copyleft. Cette licence impose que si vous utilisez cet éditeur, votre code doit utiliser la même licence. Dans le cas contraire, il est possible d'acheter une licence commerciale.

Il permet la mise en forme classique, le copier-coller d'image (base64), quelques styles.



La version testée dans cette activité est une ancienne version 4, car la version 5 modifie le fonctionnement. Il est donc important de lire la documentation et le guide de migration.

La version 5 ajoute un logo "Powered by CKEditor".

Le principe de fonctionnement est de remplacer un champ de saisie de type `textarea` par l'éditeur CKEditor. Ce dernier permet de récupérer le contenu du formulaire en HTML. Bien entendu, l'insertion de balises `<script>` ne permet pas d'injecter du code exécutable (CSRF).

L'idéal est donc de stocker ce champ comme un texte, dans une base de données, afin de garder la mise en forme de l'utilisateur.

B.1.2 Code de test

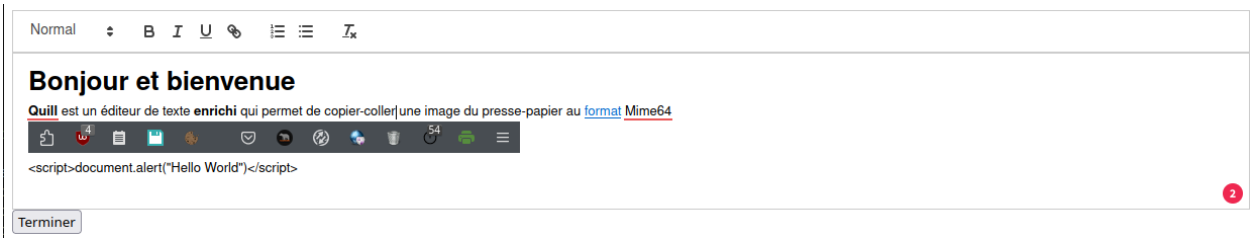
Voici un code pour tester CKEditor 4, en HTML :

```
<!doctype html>
<html lang="en">
<head>
     <meta charset="UTF-8">
     <title>CKEditor Rich Text Editor</title>

     <script src="https://cdn.ckeditor.com/4.9.2/standard/ckeditor.js"></script>
</head>
<body>
     <div><textarea name="editor1"></textarea></div>
     <div><input type="button" value="Terminer" id="btnTerminer"></div>
     <p id="resultText">ici le résultat</p>

     <script>
          CKEDITOR.replace('editor1')

          function getData() {
               //Get data written in first Editor
               let editor_data = CKEDITOR.instances['editor1'].getData()
               console.log(editor_data)
               document.getElementById("resultText").innerHTML = editor_data
          }

          let btnTerminer = document.getElementById("btnTerminer")
          btnTerminer.addEventListener("click", getData)

     </script>
</body>
</html>
```

B.2 Quill

B.2.1 Généralités

L'éditeur Quill (https://quilljs.com/) est open source et se veut plus simple. Son API modulaire permet de le personnaliser et la documentation est facilement accessible.



B.2.2 Code

Voici le code pour faire fonctionner Quill en version 2.0 :

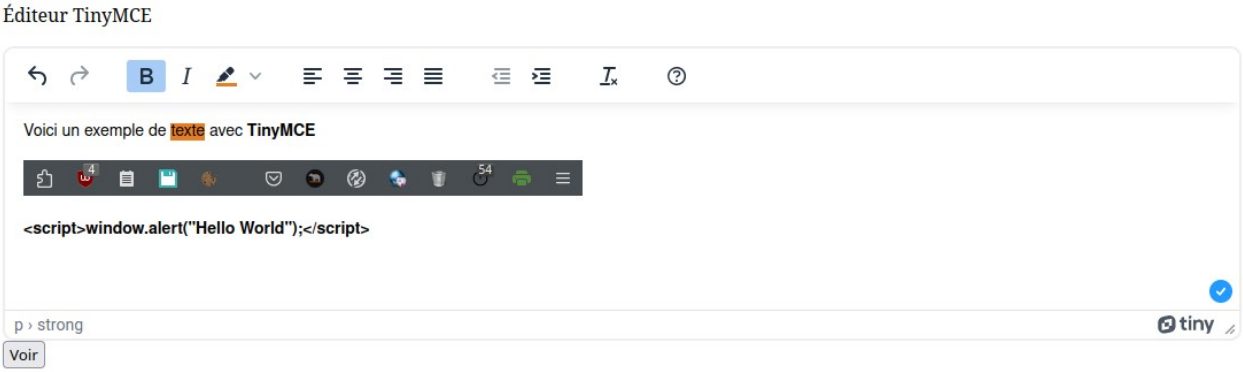
```
<!doctype html>
<html lang="en">
<head>
     <meta charset="UTF-8">
     <title>Rich Text Editor</title>

     <!-- Include stylesheet and Quill library-->
     <link href="https://cdn.jsdelivr.net/npm/quill@2.0.0-rc.3/dist/quill.snow.css" rel="stylesheet" />
     <script src="https://cdn.jsdelivr.net/npm/quill@2.0.0-rc.3/dist/quill.js"></script>

</head>
<body>
     <h1>Racontez votre histoire :</h1>
     <!-- Create the editor container -->
     <div id="editor">
       <p>Bonjour et bienvenue</p>
       <p><strong>Quill</strong> est un éditeur de texte <strong>enrichi</strong> qui permet de
copier/coller une image du presse-papier au format Mime64</p>
       <p><br></p>
     </div>
     <div><input type="button" value="Terminer" id="btnTerminer"></div>
     <p id="resultText">ici le résultat</p>

     <!-- Initialize Quill editor -->
     <script>
       var quill = new Quill('#editor', {
          theme: 'snow'
       });
     </script>

     <!-- Code to get back content from Quill -->
     <script>
          let btnTerminer = document.getElementById("btnTerminer")
          btnTerminer.addEventListener("click", () => {
               console.log(quill.getText())
               console.log(quill.root.innerHTML)
               document.getElementById("resultText").innerHTML = quill.root.innerHTML
          })
     </script>
</body>
</html>
```

B.3 TinyMCE

B.3.1 Généralités

TinyMCE est un ténor du marché et probablement le plus connu. Disponible sur https://www.tiny.cloud/get-tiny/, il est proposé en deux versions : cloud ou auto-hébergée.

Bien qu'il puisse paraître moins complet, il est très personnalisable et sa documentation est très complète.



Le concept de TinyMCE est d'avoir une partie de code pour le choix des options dans TinyMCE, puis une deuxième partie contenant l'éditeur.

B.3.2 Code

Voici le code pour utiliser TinyMCE :

```
<!DOCTYPE html>
<html>
     <head>
          <meta charset="UTF-8">
          <title>Document</title>
          <script src="https://cdnjs.cloudflare.com/ajax/libs/tinymce/6.8.3/tinymce.min.js"></script>
          <script>
          // Paramètre pour les options du module TinyMCE (format JSON)
          tinymce.init({
            selector: 'textarea#basic-example',
            height: 500,
            menubar: false,
            plugins: [
               'advlist autolink lists link image charmap print preview anchor',
               'searchreplace visualblocks code fullscreen',
               'insertdatetime media table paste code help wordcount'
            ],
            toolbar: 'undo redo | formatselect | ' +
            'bold italic backcolor | alignleft aligncenter ' +
            'alignright alignjustify | bullist numlist outdent indent | ' +
            'removeformat | help',
            content_style: 'body { font-family:Helvetica,Arial,sans-serif; font-size:14px }'
          });

          </script>
     </head>
     <body>
          <p>Éditeur TinyMCE</p>
          <textarea id="basic-example">Voici un exemple avec <strong>TinyMCE</strong></textarea>
          <input type="button" value="Voir" id="btnVoir">
          <p id="resultText">ici le résultat</p>
          <script>
               // Action de lecture du contenu du champ textarea sur clic du bouton :
               let btnVoir = document.querySelector("#btnVoir")
               btnVoir.addEventListener("click", () => {
                    // Get the HTML contents of the currently active editor
                    document.getElementById("resultText").innerHTML = tinyMCE.activeEditor.getContent()

                    // Get the raw contents of the currently active editor
                    console.log(tinyMCE.activeEditor.getContent({format : 'raw'}))

                    // Get content of a specific editor:
                    console.log(tinyMCE.get('basic-example').getContent())
               })
          </script>
     </body>
</html>
```