Installation and use of Ornement

Ornement is a software solution for **enriching text using tags**. The advantage of Ornement is that it is **easy to install and use**. The purpose of this document is to describe how to use Ornement using all the available tags. The javascript standard used is **ES6**.

Installation

Here are the different steps for installing Ornement in your project:

1. Download Ornement from its repository into the folder you want, which may be in your project.
2. That's it, you only need to complete one step to install Ornement!

Use

Demo

You can quickly view Ornement as follows:

- First you need to have Node.js installed on your PC.
- Using a terminal, install http-server (npm install http-server), then launch the server (http-server) (You can also use Live Server from VS Code if you are more comfortable with it).
- Open your browser and type http://localhost:8080.
- Go to the folder where you installed Ornement.
- Then go to Demo and open testForm.html.
- In the text box, enter either or all at once:
    - [T{style:"t2", font-style:"u"}]Introduction[/T]
    - [M{src:"https://es.wikipedia.org/static/images/icons/wikipedia.png"}]Wikipedia Logo[/M]

- [T{link:"https://fr.wikipedia.org/wiki/Ada_Lovelace"}]https://fr.wikipedia.org/wiki/Ada_Lovelace[/T]

The Engine

The engine consists of the **Ornement.js** file and the **Parsers** folder. This folder is itself made up of various **parsers** whose purpose is to **generate html** code from the Ornement tags that are used.

Integrating Ornement into your project

To integrate Ornement into your project, you need to install the folder in your project as mentioned in Installation (above).
A tag is first used in an input or textarea field **(1)**. You then need to import Ornement and retrieve the content of this field **(2)**. Finally, you pass the recovered content to the Ornement engine, which generates and returns the html tags **(3)**.

We're going to take a detailed look at how to use Ornement in Javascript (**ES6 and later versions**). I'll be using the files in the Demo folder, which are available to you:

1. Create an input or textarea field, and assign it an ID, in this case Ornement.
   ```
   <textarea id="Ornement"></textarea>
   ```
   Create a tag which will be used to store the tags obtained using Ornement, in this case I'm creating a div tag, with result as the ID.
   ```
   <div id="result">
   </div>
   ```
2. Import Ornement and retrieve the data:
   a. Import the Ornement engine as follows (adapt the file path to your situation).
   ```
   import { Ornement } from "../Ornement.js";
   ```
   b. Retrieve the data as follows.
   ```
   const textareaValue = document.getElementById("Ornement").value;
   ```
3. Generate the tags as follows:
   a. create a variable whose purpose is to be linked to the div tag with the result ID.
   ```
   const resultDiv = document.getElementById("result");
   ```
   b. Pass the recovered content (textareaValue) into Ornement, then integrate the result returned by Ornement into the result div.
   ```
   // Call the Ornement function and display the result
   resultDiv.innerHTML = Ornement(textareaValue);
   ```
   The result should now be displayed in your tag!

Feel free to use the files in the Demo folder.

Using the different tags

Here is a general example of the use of a tag, identified here by a base S (so it doesn't exist!):

- **[S{attribute1:”value1”, etc.}]Text[/S]** to add rich text, a list, code or an image.
    - **S** corresponds to the tag.
    - **[S...]** is the opening tag used.
    - **{attribute1: ‘value1’, etc.}** are the attributes of the tag.
    - **Text** is the text of the tag.
    - **[/S]** is the closing tag used.
- An escape character will be provided so that the symbols can be used freely.

/!\ between **attribute1**: and **‘value1’** no space is allowed!

The rest of this document describes the various tags that can be used, each of which can be tested using the testForm.html file.

The T tag

This tag can be used to enrich text of any length. It is used in the following way:
[T{attribute1:”value1”, etc.}]Text[/T]

The different attributes allow the tag to be used in different ways:

- **font-style:** Used to style the text. Possible values are:
    - **i**: italic, [T{font-style:"i"}]Text[/T].
    - **b**: bold, [T{font-style:"b"}]Text[/T].
    - **u**: underlined, [T{font-style:"u"}]Text[/T].
    - **s**: barré, [T{font-style:"s"}]Text[/T].
    - You can use all or some of these elements together, for example : [T{font-style:"bui"}]Text[/T].
- **link**: Inserts a link in the text, [T{link:"https://fr.wikipedia.org/wiki/Ada_Lovelace"}]https://fr.wikipedia.org/wiki/Ada_Lovelace[/T].
- **color**: Used to specify a colour in hexadecimal format only, [T{color:"#00FF00"}]Texte[/T] (will give a green text).
- **highlight-color**: Allows you to set a highlight colour in hexadecimal format only, [T{highlight-color:"#00FF00"}]Texte[/T] (will give text underlined in green).
- **style**: Used to style the text. Here are the possible values:
    - **tx**: For your text to be understood as a title, x can be from 1 to 6, [T{style:"t3"}]Sources:[/T], [T{style:"t2"}]Sources:[/T], [T{style:"t5"}]Sources:[/T].
    - sup: So that your text is in clue form, test[T{style:"sup"}]2[/T].
    - sub: To make your text superscript, test[T{style:"sub"}]2[/T].

You can combine the various attributes. For example, if you want text in red, highlighted in yellow, bold and underlined, you would use [T{color:"#FF0022", highlight-color:"#FFFF00",font-style:"ub"}]Lots of cumulative attributes[/T]. You can put spaces, after the comma, between attributes.

The L tag

This tag is used to create bulleted or numbered lists. It is used in the following way:
[L{type:"type"}]
#element1;
#element2;
…
#element3;
[/L]

Here are the criteria and rules to follow:

- 'type' must always be specified, it can take the empty type "", bullet point "b", or numbered "n". The empty type corresponds by default to the bullet point type:
    - Bullet point type:
      [L{type:"b"}]
      #élément 1;
      #élément 2;
      #élément 3;
      [/L]

      ou

      [L{type:""}]
      #élément 1;
      #élément 2;
      #élément 3;
      [/L]

    - Numbererd type:
      [L{type:"n"}]
      #élément 1;
      #élément 2;
      #élément 3;
      [/L]

- A line feed is required between the opening tag and the first element of the list, between each element of the list, and between the last element of the list and the closing tag.
- each list item starts with a diez "#" and ends with a semicolon ";", between the diez and the item or between the item and the semicolon, there can be many spaces.

The M tag

The M tag can be used to include online media, such as pictures, videos or audio. It is used as follows:
[M{src:”lien_media”}]Text[/M]

Here are the characteristics of this tag:

- The src attribute is used to enter the media address, which must be on the Internet.
- The text between the M tags is not compulsory, but it will help you remember the image you have set in edit mode.
- The following types of media are authorized:
    - 'jpg', 'jpeg', 'png', 'gif', 'bmp', ‘webp’ for pictures.
    - 'mp3', 'wav', 'ogg', 'aac' for audios.
    - 'mp4', 'webm', 'avi', 'mkv'for videos.

The C tag

The C tag is used to include stylized code. It has no attributes. It is used as follows:

[C]
Code
[/C]

Skip a line after the opening tag, and skip a line at the end of the written code.