WProofreader initialization options

There are several options letting you initialize WProofreader functionality in your web app, namely using:

- **autoSearch** option (WEBSPELLCHECKER_CONFIG and inline data attributes);
- **init()** method;
- **autocreate** (inline data attributes).

The WEBSPELLCHECKER_CONFIG (further CONFIG) can be added in a script or loaded from the file as mentioned in Get Started with WProofreader Server (autoSearch) guide for Server version and Get Started with WProofreader Cloud (autoSearch).

After you have initialized WProofreader, you can customize the above-mentioned settings using WProofreader API.

Below you can find the samples showing how different WProofreader initialization approaches work depending on the integration and expected launch behavior.

1. Initializing using autoSearch

The **autoSearch** feature enables detecting new editable fields on the page and proofreading the text they contain automatically in focus. If you choose this option, WProofreader is enabled in the selected WYSIWYG editor or HTML editable control, and no additional actions are required.

- autoSearch is useful when your web page has multiple controls, as you can add one script which is enabled in the areas in focus automatically compared with creating an init() method for each control. From a performance point of view, autoSearch also has better performance as the grammar and spelling and style are checked only in active area compared with checking in all controls like in init().
- autoSearch for a dynamic page where some fields can be hidden or added at a certain point is enabled automatically and releases the memory, helping to avoid possible memory leaks.

Initialization using inline attributes is suitable and useful if you want to have a single script with basic option definitions.

Example 1.1. Initializing using CONFIG

This is an example of the WProofreader initialization with the autoSearch functionality turned on. All the configuration options are defined using CONFIG.

```
<script>
    window.WEBSPELLCHECKER_CONFIG = {
                autoSearch:true,
                lang: 'uk_UA',
                ...
        };
</script>

<script type="text/javascript" src="http(s)://your_host_name/wscservice/wscbundle/wscbundle.js"></script>
```

Example 1.2. Initializing using inline data attributes

This is an example of the WProofreader initialization with the autoSearch functionality turned on. All the configuration options are defined using inline data attributes. In this case, the naming of the options are formed as follows: **data-wsc-option_name=”value”.** For details, refer to the list of available options in WProofreader API documentation.

```
<script
        data-wsc-autosearch="true"
        data-wsc-lang="uk_UA"
        ...
    src="https://your_host_name/wscservice/wscbundle/wscbundle.js">
</script>
```

⚠ When initializing using data attributes, a number of API parameters with **array** or **number** type are not supported. Thus, you cannot modify such parameters as **actionItems**, **suggestionsCount**, and **moreSuggestionsCount** described further in this section. When initializing WProofreader using CONFIG, all options are supported.

2. Initializing using init() method

The **init() method** is a reasonable choice when you clearly know in which control you want to initialize WProofreader. More than that you are aware of your page controls, and additional controls do not appear dynamically.

- If your web page has several editable elements where you want to initialize WProofreader, as soon as the user loads the page, WProofreader is enabled automatically. It is not required to select the area to enable grammar and spelling check in it compared with **autoSearch** method.
- With dynamic web page load using **init()** method, administrator needs to be monitor when it's necessary to start WProofreader.

Example 2.1. Initializing in HTML element using init() method

This is an example of WProofreader initialization in HTML contenteditable element (**div**) using **init()** method. The configuration options are specified directly in the init() function.

```
<script type="text/javascript" src="http(s)://your_host_name/wscservice/wscbundle/wscbundle.js"></script>

<div contenteditable id="container1">
        <p>This sampl text is aimed at demonstrating the work of WProofreader in a contenteditable div element.<
/p>
</div>

<script>
    var instance1 = WEBSPELLCHECKER.init({
                container: document.getElementById("container1"),
                lang: 'uk_UA',
                ...
        });
</script>
```

Example 2.2. Initializating in multiple HTML elements using init() method

This is an example of WProofreader initialization in HTML contenteditable element **(div)** and HTML editable element **(textarea)** using **init()** method. The configuration options for both controls are defined separately in a single CONFIG.

```
<script>
        window.WEBSPELLCHECKER_CONFIG = {
                lang: 'uk_UA',
                ...
    };
</script>

<script type="text/javascript" src="http(s)://your_host_name/wscservice/wscbundle/wscbundle.js"></script>

<div contenteditable id="container1">
        <p>This sampl text is aimed at demonstrating the work of WProofreader in a contenteditable div element.<
/p>
</div>

<script>
    var instance1 = WEBSPELLCHECKER.init({
                container: document.getElementById("container1"),
        });
</script>

<textarea id="container2" type="text">This sampl text is aimed at demonstrating the work of WProofreader in a
textarea textform element.</textarea>

<script>
    var instance2 = WEBSPELLCHECKER.init({
                container: document.getElementById("container2"),
        });
</script>
```

Example 2.3. Initializing in WYSIWYG editors using init() method

There is an option to explicitly initialize WProofreader in such rich text editors as **Froala Editor 3**, **CKEditor 4**, and **CKEditor 5** using **init()** method. However, it is strongly recommended to use this method with **autoSearch** and **autoDestroy** options in CONFIG.

Turned on/enabled **autoDestroy** parameter will be monitoring the state of the WEBSPELLCHECKER instance and handling its destroy after removal (deleted or hidden) of an editable container from the page. At the same time enabled **autoSearch** parameter will restore an instance when it is needed. It can be useful for example, when switching to the code editing mode in the editor, WEBSPELLCHECKER instance must be deleted and then restored after returning back to the editor. Thus, it is the autoSearch that restores it.

Initializing WProofreader using init() in CKEditor 4

```
<!-- Include the WEBSPELLCHECKER_CONFIG variable. -->
<script>
    window.WEBSPELLCHECKER_CONFIG = {
                autoSearch: true,
                autoDestroy: true,
                ...
    };
</script>

<script type="text/javascript" src="http(s)://host_name/wscservice/wscbundle/wscbundle.js"></script>

<!-- Use this path for the Cloud-based version
<script type="text/javascript" src="https://svc.webspellchecker.net/spellcheck31/wscbundle/wscbundle.js"><
/script>
-->

<div id="ckeditor4-editor">
    <p>These are an examples of a sentences with two mispelled words and grammar problems. Just type text with
misspelling to see how it works.</p>
</div>

<script>
        CKEDITOR.disableAutoInline = true;

        CKEDITOR.on('instanceReady', function(event) {
                var editor = event.editor;

            WEBSPELLCHECKER.init({
                        //catch both iframe and inline modes
                container: editor.window.getFrame() ? editor.window.getFrame().$ : editor.element.$
                });
        });

        CKEDITOR.replace('ckeditor4-editor', {});
</script>
```

Initializing WProofreader using init() in CKEditor 5

```
<!-- Include the WEBSPELLCHECKER_CONFIG variable. -->
<script>
    window.WEBSPELLCHECKER_CONFIG = {
                autoSearch: true,
                autoDestroy: true,
                ...
    };
</script>

<script type="text/javascript" src="https://your_host_name/wscservice/wscbundle/wscbundle.js"></script>

<div id="ckeditor5-editor">
        <p>These are an examples of a sentences with two mispelled words and grammar problems. Just type text
with misspelling to see how it works.</p>
</div>

<script>
        ClassicEditor
            .create(document.querySelector('#ckeditor5-editor'))
            .then(editor => {
                WEBSPELLCHECKER.init({
                    container: editor.ui._editableElements.get('main')
                });
            });
</script>
```

Initializing WProofreader using init() in Froala Editor

```
<!-- Include the WEBSPELLCHECKER_CONFIG variable. -->
<script>
    window.WEBSPELLCHECKER_CONFIG = {
                autoSearch: true,
                autoDestroy: true,
                ...
    };
</script>

<script type="text/javascript" src="https://your_host_name/wscservice/wscbundle/wscbundle.js"></script>

<div id="froala-editor">
    <p>These are an examples of a sentences with two mispelled words and grammar problems. Just type text with
misspelling to see how it works.</p>
</div>

<script>
    new FroalaEditor('#froala-editor', {
                iframe: true,
        events: {
            'initialized': function() {
                WEBSPELLCHECKER.init({
                    container: this.$iframe ? this.$iframe[0] : this.el
                });
            }
        }
    });
</script>
```

3. Initializing using data-wsc-autocreate

This approach is similar to initializing WProofreader using init() method.

Example 3.1. Initializing in HTML elements using data-wsc-autocreate

This is an example of WProofreader initialization in HTML contenteditable element **(div)** and HTML editable element **(textarea)** using **data-wsc-autocreate="true"**. The configuration options are to be defined separately in CONFIG.

Even though we have a CONFIG where we defined Ukrainian as a language for check, the language for textarea will be obtained from the data-wsc-lang="es_ES" setting. It will have higher priority and, thus, will rewrite the CONFIG settings in the sample below.

```
<script>
    window.WEBSPELLCHECKER_CONFIG = {
        lang:’uk_UA’,
    };
</script>

<script type="text/javascript" src="http(s)://your_host_name/wscservice/wscbundle/wscbundle.js"></script>

<div contenteditable data-wsc-autocreate="true">
        <p>This sampl text is aimed at demonstrating the work of WProofreader in a plain textarea element.</p>
</div>

<textarea type="text" data-wsc-autocreate="true" data-wsc-lang="es_ES">This sampl text is aimed at
demonstrating the work of WProofreader in a textarea textform element.</textarea>
```