Using Your Own Document Styles

As a developer, Textbox.io enables you to take fine-grained control over the way content is presented, as well as the styles that users are able to select from the styles dropdown menu.

Injecting your own CSS files

> In Classic Editing Mode, the textbox.io editor document is contained within an `iframe` element. This means that any styles or stylesheets declared on your **host** page **are not** inherited by the editor. These need to be explicitly declared as references in the editor configuration.

CSS and styles are configured per-instance as part of the Textbox.io configuration object

You can declare your own stylesheets using the `stylesheets` property or inject CSS directly using the `documentStyles` property:

```
var configCSS = {
    css : {
        stylesheets : ['http://www.example.com/mycss.css', 'anotherfile.css'], // an array of CSS file URLs
        documentStyles : 'body { background:red; }' // a string of CSS
    }
};
```

Special cases: Table cells & List Items

The list of items shown on the styles drop-down menu are dependent on the current editor selection. Custom block styles declared for table cells or list items will only become visible when the editor selection exists on those elements as these styles are only valid CSS within a table or list structure. In this way Textbox.io prevents users from creating invalid HTML.

Custom styles for table cells will only appear when the selection is on a table

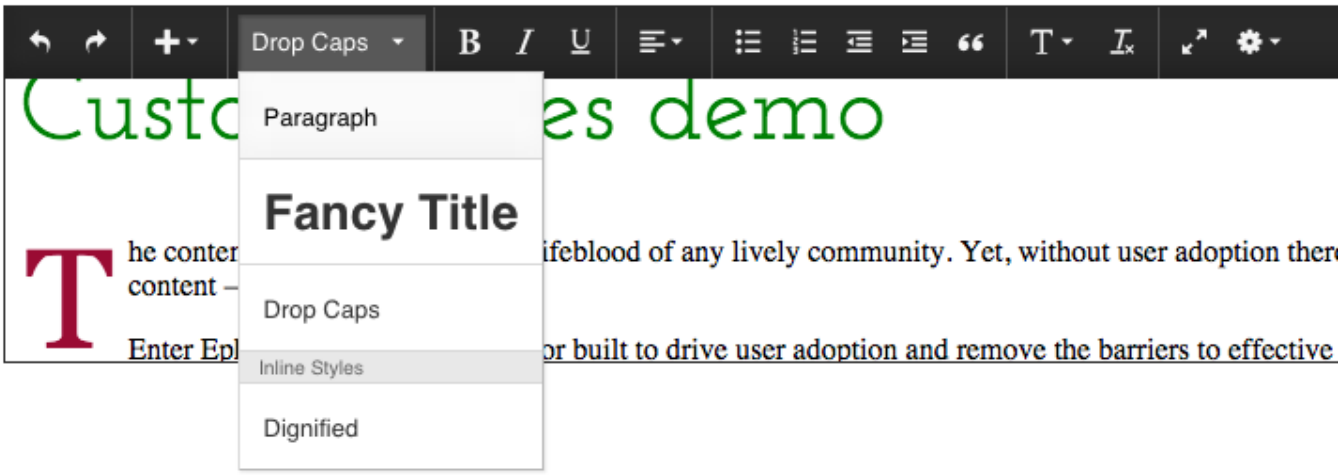
Example: custom CSS

The example below shows the editor using a highly customized CSS file:

Customizing the styles drop-down menu

The items shown in Textbox.io's styles drop-down menu can be customized to suit the requirements of your application. There configuration approaches available are:

- A fixed list of custom styles defined per-editor using the the editor CSS configuration object.
- The editor inspects configured content CSS to optionally show additional styles on the menu *(new in version 2.4.1)*

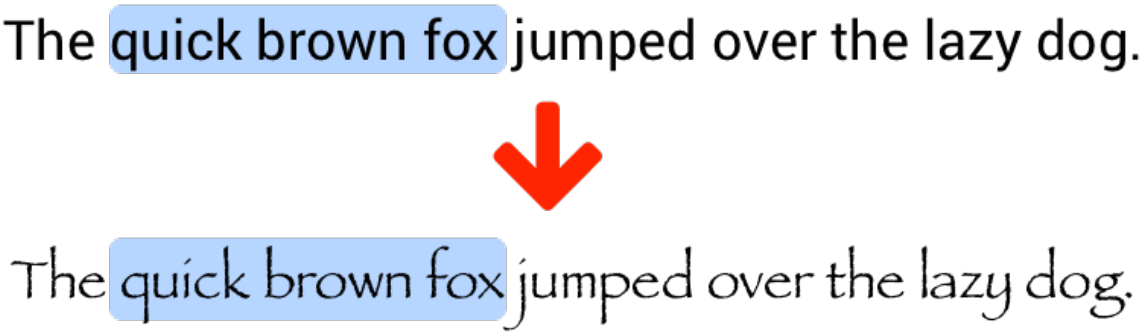


The Textbox.io styles drop-down menu

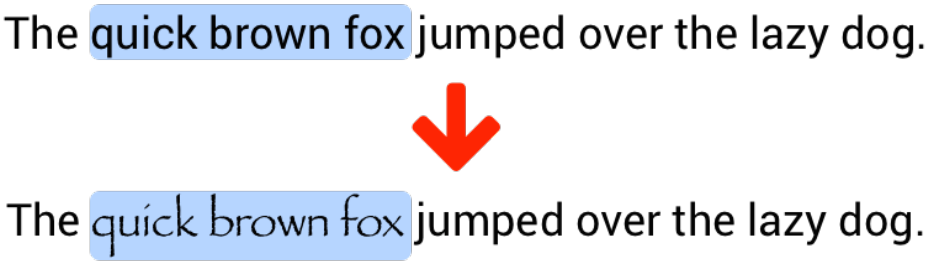
Overview: Block vs. Inline styles

Before attempting to create your own custom styles, it is worthwhile understanding the difference between *block style transformations*, and *inline transformations.*

- When a **block transformation** is applied to a selection, the entire block encasing the selection (such as a paragraph or heading) is transformed:



- When an **inline transformation** is applied, to a selection, only the selection itself (and no surrounding content) is transformed:



In its stock configuration, the drop-down styles menu contains **only block style transformations** - inline style changes can be made via the font drop-down menu.

A detailed explanation of transformation behaviour is included with the styles configuration object documentation.

Defining Styles in the configuration object

The styles drop-down can be specified in the editor styles configuration object.

The following is an example of custom styles:

- *"fancy title"* is an example of a *block style*. Similarly, if the rules had defined `h2.title`, `h3.title`, or `p.title`, this transformation would take place at a block level.
- *"dignified"* is an example of an *inline style*, where the prefixing element has been removed from the definition.

Note the differences in the presentation of these styles in the drop-down menu declaration compared to the CSS declaration file.

Defining styles in the content CSS

This feature was introduced in Textbox.io release 2.4.1

The editor inspects all content styles and can be configured to add style entries automatically. This is useful in scenarios where the editor configuration is fixed across an entire CMS platform but the content stylesheet is set by a template. With this configuration, the styles dropdown can change with the template rather than requiring configuration changes.

Showing or hiding styles

When specifying CSS for use in an instance of the editor, you may not necessarily want every style to appear in the styles drop-down combo box. Using specific CSS attributes you can hide or display specified CSS elements in the styles drop-down.

This process is controlled by the showDocumentStyles configuration option, disabled by default, which allows for detailed control over the process:

- When disabled, only styles explicitly set to visible are shown. This can be useful when a complete stylesheet is used in the editor content and showing them all would result in a confusing drop-down.
- When enabled, all styles are shown unless explicitly set to hidden. This is useful when a dedicated stylesheet is created for the editor content.

Custom styles are subject to the same restrictions as rules defined in the configuration object. Any rules that are not valid under those restrictions are ignored regardless of their visibility attribute.

CSS definition attributes

In order to leverage the document stylesheet to achieve this feature, the attributes used must be completely valid CSS. To this end, the editor looks for rules that are targeted at an element attribute (`ephox-data`) which the editor will never set and therefore the rule should never activate in normal use. Using the `visibility` and `content` properties, both whether a style is shown and the text shown in the menu can be controlled.

```
/* never shown in the drop-down, regardless of configuration */
h1.blue {
  color: blue
}
h1.blue[ephox-data] {
  visibility: hidden;
}

/* always shown in the drop-down, regardless of configuration */
h1.red {
  color: red
}
h1.red[ephox-data] {
  visibility: visible;
  content: "red heading";
}

/* Only shown in the drop-down when showDocumentStyles is true */
h1.green {
  color: green;
}
h1.green[ephox-data] {
  content: "green heading";
}
```

Content style entries defined in this way are added *below* any rules defined in the configuration object, which allows either a mixture of fixed and dynamic styles or (if configured with an empty list) a completely dynamic style drop-down.

The ability of the editor to inspect the content CSS is dependent on CORS. If an external content stylesheet is on a different domain to the editor page, the CSS request must return CORS headers otherwise the browser blocks scripts from inspecting the stylesheet.

Custom style examples

showDocumentStyles disabled

This example replicates the configuration object example above, but adds rules to the style drop-down using the content CSS rather than using the configuration object. In this case `showDocumentStyles` is false, so only the style explicitly configured is added.

showDocumentStyles enabled

This example inverts the `showDocumentStyles` config from above. In this mode, all styles are shown on the menu except the one that is hidden through a CSS property. The configuration object is an empty array indicating only document styles should be shown.