Chapter 45: execCommand and contenteditable

commandId	value
⋮ **Inline formatting commands**	
backColor	Color value String
bold	
createLink	URL String
fontName	Font family name
fontSize	"1", "2", "3", "4", "5", "6", "7"
foreColor	Color value String
strikeThrough	
superscript	
unlink	
⋮ **Block formatting commands**	
delete	
formatBlock	"address", "dd", "div", "dt", "h1", "h2", "h3", "h4", "h5", "h6", "p", "pre"
forwardDelete	
insertHorizontalRule	
insertHTML	HTML String
insertImage	URL String
insertLineBreak	
insertOrderedList	
insertParagraph	
insertText	Text string
insertUnorderedList	
justifyCenter	
justifyFull	
justifyLeft	
justifyRight	
outdent	
⋮ **Clipboard commands**	
copy	Currently Selected String
cut	Currently Selected String
paste	
⋮ **Miscellaneous commands**	
defaultParagraphSeparator	
redo	
selectAll	
styleWithCSS	
undo	
useCSS	

Section 45.1: Listening to Changes of contenteditable

Events that work with most form elements (e.g., `change`, `keydown`, `keyup`, `keypress`) do not work with `contenteditable`.

Instead, you can listen to changes of `contenteditable` contents with the `input` event. Assuming `contenteditableHtmlElement` is a JS DOM object that is `contenteditable`:

```
contenteditableHtmlElement.addEventListener("input", function() {
    console.log("contenteditable element changed");
});
```

Section 45.2: Getting started

The HTML attribute `contenteditable` provides a simple way to turn a HTML element into a user-editable area

```
<div contenteditable>You can <b>edit</b> me!</div>
```

Native Rich-Text editing

Using **JavaScript** and execCommandW3C you can additionally pass more editing features to the currently focused `contenteditable` element (specifically at the caret position or selection).

The `execCommand` function method accepts 3 arguments

```
document.execCommand(commandId, showUI, value)
```

- `commandId` String. from the list of available ***commandId***s
  (see: ***Parameters***→*commandId*)
- `showUI` Boolean (not implemented. Use **`false`**)
- `value` String If a command expects a command-related String ***value***, otherwise `""`.
  (see: ***Parameters***→*value*)

Example using the `"bold"` **command** and `"formatBlock"` (where a **value** is expected):

```
document.execCommand("bold", false, "");          // Make selected text bold
document.execCommand("formatBlock", false, "H2"); // Make selected text Block-level <h2>
```

Quick Start Example:

```
<button data-edit="bold"><b>B</b></button>
<button data-edit="italic"><i>I</i></button>
<button data-edit="formatBlock:p">P</button>
<button data-edit="formatBlock:H1">H1</button>
<button data-edit="insertUnorderedList">UL</button>
<button data-edit="justifyLeft">&#8676;</button>
<button data-edit="justifyRight">&#8677;</button>
<button data-edit="removeFormat">&times;</button>

<div contenteditable><p>Edit me!</p></div>

<script>
[].forEach.call(document.querySelectorAll("[data-edit]"), function(btn) {
  btn.addEventListener("click", edit, false);
});
```

```
function edit(event) {
  event.preventDefault();
  var cmd_val = this.dataset.edit.split(":");
  document.execCommand(cmd_val[0], false, cmd_val[1]);
}
<script>
```

jsFiddle demo
Basic Rich-Text editor example (Modern browsers)

Final thoughts
Even being present for a long time (IE6), implementations and behaviors of `execCommand` vary from browser to browser making "building a Fully-featured and cross-browser compatible WYSIWYG editor" a hard task to any experienced JavaScript developer.
Even if not yet fully standardized you can expect pretty decent results on the newer browsers like **Chrome, Firefox, Edge**. If you need *better* support for other browsers and more features like HTMLTable editing etc. a rule of thumbs is to look for an **already existent** and robust **Rich-Text** editor.

Section 45.3: Copy to clipboard from textarea using execCommand("copy")

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
    <textarea id="content"></textarea>
    <input type="button" id="copyID" value="Copy" />
    <script type="text/javascript">
        var button = document.getElementById("copyID"),
            input = document.getElementById("content");

        button.addEventListener("click", function(event) {
            event.preventDefault();
            input.select();
            document.execCommand("copy");
        });
    </script>
</body>
</html>
```

`document.execCommand("copy")` copies the current selection to the clipboard

Section 45.4: Formatting

Users can add formatting to `contenteditable` documents or elements using their browser's features, such as common keyboard shortcuts for formatting (`Ctrl-B` for **bold**, `Ctrl-I` for *italic*, etc.) or by dragging and dropping images, links, or markup from the clipboard.

Additionally, developers can use JavaScript to apply formatting to the current selection (highlighted text).

```
document.execCommand('bold', false, null); // toggles bold formatting
document.execCommand('italic', false, null); // toggles italic formatting
```

```
document.execCommand('underline', false, null); // toggles underline
```