A WYSIWYG Framework

by

Johannes-Lukas Bombach

Submitted to the Fachbereich Informatik, Kommunikation und Wirtschaft
in partial fulfillment of the requirements for the degree of

Master of Science

at the

HOCHSCHULE FÃIJR TECHNIK UND WIRTSCHAFT BERLIN

August 2015

Author .....
Fachbereich Informatik, Kommunikation und Wirtschaft
August 26, 2015

Certified by .....
Prof. Dr. Debora Weber-Wulff
Associate Professor
Thesis Supervisor

Accepted by .....
???
Chairman, Department Committee on Graduate Theses

A WYSIWYG Framework

by

Johannes-Lukas Bombach

Submitted to the Fachbereich Informatik, Kommunikation und Wirtschaft
on August 26, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Browsers do not offer native elements that allow for WYSIWYG text editing. There are third-party libraries that emulate these elements by utilizing the `contenteditable`-attribute. However, the API enabled by `contenteditable` is limited and unstable. Bugs and unwanted behavior can only be worked around and not fixed. The library âĂđTypeâĂIJ demonstrates that WYSIWYG editing can be achieved without requiring the `contenteditable` attribute, thus solving many problems contemporary WYSIWYG editor libraries have.

Thesis Supervisor: Prof. Dr. Debora Weber-Wulff
Title: Associate Professor

Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements.

Contents

List of Figures

List of Tables

Chapter 1

DOM manipulation

In October 1998 the World Wide Web Consortium (W3C) published the "Document Object Model (DOM) Level 1 Specification". This specification includes an API on how to alter DOM nodes and the document's tree[1]. This provided a standardized way for changing a website's contents. With the implementations of Netscape's JavaScript and Microsoft's JScript this API has been made accessible to developers.

1.1 HTML Editing APIs

In July 2000, with the release of Internet Explorer 5.5, Microsoft introduced the contentEditable and designMode IDL attributes along with the contenteditable content attribute[2][3]. These attributes were not standardized and not part of the W3C DOM specifications.

By settings these settings to "true" (`contenteditable` / `contentEditable`) or

[1]http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html, last checked on 07/10/2015
[2]https://msdn.microsoft.com/en-us/library/ms533720(v=vs.85).aspx, last checked on 07/10/2015
[3]https://msdn.microsoft.com/en-us/library/ms537837(VS.85).aspx, last checked on 07/10/2015

Attribute	Type	Can be set to	Possible values
designMode	IDL attribute	Document	"on", "off"
contentEditable	IDL attribute	Specific HTMLElements	boolean, "true", "false", "inherit"
contenteditable	content attribute	Specific HTMLElements	empty string, "true", "false"

Table 1.1: Editing API attributes

"on" (`designMode`), Internet Explorer switches the affected elements and their children to an editing mode. In editing mode it is possible to

1. Let the user interactively click on and type inside text elements
2. Execute "commands" via JScript and JavaScript

Fetching user inputs (clicking on elements, accepting keyboard input and modifying text nodes) is handled entirely by the browser. No further scripting is necessary other than setting the mentioned attributes on elements. This behavior is inherited by child elements.

In editing mode, calling the method document.execCommand will format the currently selected text depending on the parameters passed. Calling `document.execCommand('bold', false, null)` will wrap the currently selected text in `<b>` tags. `document.execCommand('createLink' false, 'http://google.com/')` will wrap the selected text in a link to google.com. However, this command will be ignored, if the current selection is not contained by an element in editing mode.

While `designMode` can only be applied to the entire document, `contentEditable` and `contenteditable` attributes can be applied to a subset of HTML elements as described on Microsoft's Developer Network (MSDN) online documentation[2].

With the release of Internet Explorer 5.5 and the introduction of editing capabilities, Microsoft released a sparse documentation describing only the availability and the before-mentioned element restrictions of these attributes[2].

According to Mark Pilgrim, author of the âĂđDive intoâĂIJ book series and contributor to the the Web Hypertext Application Technology Working Group (WHATWG), Microsoft did not state a specific purpose for its editing API, but, its first use-case has been rich-text editing[3].

In March 2003, the Mozilla Foundation introduced an implementation of Microsoft's designMode, named Midas, for their release of Mozilla 1.3. Mozilla names this âĂđrich-text editing supportâĂIJ on the Mozilla Developer Network (MDN)[4]. In June 2008, Mozilla added support for contentEditable IDL and contenteditable content attributes with Firefox 3.

MozillaâĂŹs editing API resembles the API implemented for Internet Explorer, however, there are still differences (compare [5][6]). Most notably, Microsoft and Mozilla differ in the commands provided to pass to document.execCommand[6][7] and the markup generated by invoking commands[8]. In fact, Mozilla only provides commands dedicated to text editing while Microsoft offers a way to access lower-level browser components (like the browserâĂŹs cache) using execCommand. This may show, that even though rich-text editing was its first use case and Mozilla implemented it naming it that, this editing API was not originally intended to be used as such.

In March 2008, Apple released Safari 3.1[8a] including full support for contentEditable and designMode[9], followed by Opera Software in June 2009[XXX] providing full support in Opera 9[9a]. MDN lists full support in Google Chrome since version 4[11], released in January 2010[12].

Around the year 2003[13] the first JavaScript libraries emerged that made use of MicrosoftâĂŹs and MozillaâĂŹs editing mode to offer rich-text editing in the browser. Usually these libraries were released as user interface components (text fields) with inherent rich-text functionality and were only partly customizable.

In May 2003 and March 2004 versions 1.0 of âĂđFCKEditorâĂIJ and âĂđTinyMCEâĂIJ have been released as open source projects. These projects are still being maintained and remain among the most popular choices for incorporating rich-text editing in web-based projects.

Appendix A

Tables

Table A.1: Armadillos

Armadillos	are
our	friends

Appendix B

Figures

Figure B-1: Armadillo slaying lawyer.

Figure B-2: Armadillo eradicating national debt.

Bibliography