

WYSIWYG

Table of contents

1	Why WYSIWYG is good	2
1.1	Editing without WYSIWYG.....	2
1.2	WYSIWYG hides complexity from the user.....	2
2	... and why WYSIWYG can be bad as well.....	2

1. Why WYSIWYG is good ...

WYSIWYG (pronounce "vee-zee-weeg") stands for "What you see is what you get". To make most people who's personal experience with computers does not at least date back into the 80ies, a little history lesson is needed here. In case you have even used punch cards or you are a *IX person who knows why vi is considered a visual editor (compared to ed), this section will not contain a lot of news for you and you can safely skip to the next section. Anyone who grew up with windows and mice read on.

1.1. Editing without WYSIWYG

An electronic document - for example an HTML page - is nothing but a file on your system which contains a character stream. You can edit an HTML page with a simple text editor such as vi or Notepad. In that case, you have to put HTML tags into your text and ensure yourself that the resulting file is valid and can be rendered by the browser.

This has nothing to do with WYSIWYG at all, as what you see on your screen while editing looks entirely different than what you will see later in your browser when the HTML file is rendered.

While many people might have written their first HTML pages in a text editor, today it's hard to imagine that early word processors worked the same way. You could not just select a word and hit a button to make the word bold and immediately see the result, but you had to put control characters into your text and could only see the result on paper after you printed the text to paper. In the early days of word processing, the software did not even understand the control characters but just sent them to the printer. Introducing printer independent control characters, which were then mapped to the specific escape codes for a specific make and model of a printer was already the first step of abstraction, which was considered an innovation when it was introduced.

Early word processors operated in text only mode which means that you cannot display different fonts, font sizes or even bold or italic script on the screen at all. Later there have been graphic cards that were able to display attributes such as bold or italic type, but it was never possible to display different font sizes in text mode. Some word processors tried to compensate this by using colors.

1.2. WYSIWYG hides complexity from the user

The introduction of graphical user interfaces probably was the single most important enabler to WYSIWYG word processors as well as any other kind of WYSIWYG editors, for example for spreadsheets, presentation slides or illustrations.

With WYSIWYG (together with some other technologies such as Postscript or TrueType fonts) it was possible to display a text exactly the way it will be printed later. You don't see any control characters any more on your screen (though they are still in the character stream) but you can select a part of the text and assign a font, font size or attribute such as bold or italics. And afterwards, you get an immediate visual feedback of what you have done and you can decide if this is really what you intended to do.

WYSIWYG has two obvious advantages:

- The user does not have to learn any control characters or tags, but he or she can manipulate the document through clicking on intuitive buttons and other GUI widgets, for example a drop down list of available fonts.
- You don't have to imagine how your document will look like later but you can see it building up in front of your eyes. For example, you can type a text first and then mark and format the headlines. Editing a large text document or even a graphical document without WYSIWYG requires a lot of imagination or a lot of experience with similar document types to avoid having too many iterations of putting something in, then having to render it for preview and then going back to the source code.

The introduction of WYSIWYG word processors led to a whole wave of WYSIWYG tools. It became next to impossible to sell any new application that could not claim it was WYSIWYG. Besides word processors, it was HTML editors and page layout systems (DTP) which became increasingly popular.

2. ... and why WYSIWYG can be bad as well

If you are looking for an example of the limits of WYSIWYG, take a look at the idea of "visual programming".

Some of the most powerful software companies as well as some smaller innovative companies have tried to deliver on the promise that it will be possible to replace writing of source code with clicking together components visually using the mouse. Aside from very specific aspects such as GUI design and linking data to a GUI nobody ever really made it.

Depending on who you ask and depending on your personal point of view, you may pick from one of these two explanations:

- It cannot be done.
- It does not make sense.

The idea of visual programming is a bit like trying not to write a letter or a novel character by character but by choosing text blocks from a library. It can certainly be done but you are very limited in the thoughts that you can express. If you try to express a thought that is new and has never been expressed with this system before, you cannot.

Taking it back on a technical level, there is a danger that indifferent application of the WYSIWYG idea leads to either oversimplifying things or having to build a GUI that is overly complex and not intuitive at all.