Notes per slide

A history of markup languages

A summary of some of the more obvious bits of the history of document markup.

And only hitting the "high spots" for what I am talking about.

However, the github repository has the source for these slides, and also a set of extended notes with links. I'll give the URL again at the end.

Timeline

I'm ignoring anything that isn't just text (so, music, mathematics, diagrams, bibliographies, indices, etc.).

Even so, this is clearly not all of the text markup formats there are, but hopefully its a good survey.

There's a lot to cover, even so.

What's interesting, though, is that almost everything named is still in use, in one form or another.

Note: I'm not going to follow a strict linear sequence in time, but instead work partially by topic.

The types of markup

Presentational: how the text should be presented, e.g., as a man page, on a screen, or on a typeset page.

Even at the beginning of our timeline, people had access to typesetters, and wanted to drive them.

Centering, right justification and so on are laborious, so worth automating.

And presentation on different displays (or even line printers) may use different techniques to produce effects such as bolding.

Semantic: marking up the meaning of the text.

One of the important early realisations was that even presentation benefits from some degree of semantics - i.e., "heading", not "font X at size Y in bold".

Indexing and creation of references is another sort of semantic markup of presentationally marked documents.

But it can also be important to mark up the meaning of text for its own sake.

Subcategories, mainly of Presentational

Lightweight markup: simple to type, and hopefully easier to read. Hence also

Programmable markup: markup with what is (essentially) a programming language (wikipedia calls this "procedural" markup). For instance, TeX.

Obviously a markup may span categories.

1964: RUNOFF

1964 RUNOFF *Presentational*

Jerome H. Saltzer for CTSS (Compatible Time Sharing System)

Commands starting with a dot in the first column.

Commands could be abbreviated, e.g., `.ls` instead of `.list`, and `.le;` instead of `list element;`.

Inline commands shift the "case", for instance in and out of bold case.

Ported to BCPL and Multics. Ancestor to roff and thus, ultimately, all of the roff family.

In the 1980s/1990s I used Digital Standard Runoff, also a direct descendant.

This example is (more or less) from the original TYPSET/RUNOFF documentation.

1969: GML and 1986: SGML

1969 GML, 1986 SGML *Semantic* and *"meta"* (DTDs)

GML (Charles Goldfarb, Edward Mosher, Raymond Lorie) at IBM.

[Standard] Generalised Markup Language.

The example is actually SGML. It is transcribed from Figure 3 of the paper named. The ellipses are mine.

The GML starter set was a set of macros for IBM Script.

A mechanism for *describing* markup languages. Use of the DTD.

Sensibly, SGML came with a "starter set" drafted by Joan Smith and Janet Vandore.

Note how SGML allowed the definition of elements that were implicitly closed by another element -e.g., <li> and <p>

- <td> is the document title
- <ad> is an address, <al> an address line
- <ab> is the abstract
- <ci> indicates a citation, which rendered as italics in the resulting paper.
- <ref> marks up a Reference, collected for the section at the end of the document.
- &SGML is an "entity reference" that expands to 'Standard Generalized Markup Language' - we're familiar with things like é from HTML.

Not in the shorter version

1969: GML / 1986: SGML

1969 GML, 1986 SGML *Semantic* and *"meta"* (DTDs)

1969 GML (Charles Goldfarb, Edward Mosher, Raymond Lorie - note the initials of the surnames) at IBM.

1986 [Standard] Generalised Markup Language.

A mechanism for *describing* markup languages.

SGML uses DTDs (Document Type Definitions) to describe the set of markup declarations that form a *document type* (e.g., SGML itself, XML, HTML).

Shown is a DTD fragment for defining a simple list, and an example of the list structure described.

Sensibly, SGML came with a "starter set" drafted by Joan Smith and Janet Vandore.

SGML allows the definition of elements that were implicitly closed by another element - e.g., <li> and <p> in HTML.

In our example:

```
<!ELEMENT list - - (item)+ >
```

- The element being defined is `list`.
- The two hyphens indicate that both the start tag `<list>` and the end tag `</list>` for this element are required.
- The `+` means that there must be "at least one `<item>` element".

In:

```
<!ELEMENT item O O (#PCDATA, (list)*)  >
```

- The two `O` ("oh", not "zero") characters mean that both the start and end tags can be omitted.
- The end of the specification tells us that an `item` may contain `PCDATA` (text) or zero or more `list` elements.

1997: XML

1997 XML (Extensible Markup Language) *Semantic*.

XML was compiled by a working group of eleven members,[30] supported by a (roughly) 150-member Interest Group.

No example because there is no "default" XML - a schema is needed.

XML was compiled by a working group of eleven members, supported by a (roughly) 150-member Interest Group.

It's a simpler subset of SGML, which makes parsers easier to write.

Other SGML based tools (TEI, Docbook, HTML itself) have generally moved towards using XML rather than SGML in their specification.

Not in the shorter version

1970s: roff, nroff, troff, groff

1970s *roff *Presentational*. Still in use (as 1990: groff)

Started as a transliteration of the BCPL version of runoff, for UNIX, around 1970.

The example is a (fake) man page, using the `man` macro package from Lars Wirzenius' Writing manual pages

- .TH = title
- .SH = sub-heading
- .B = bold
- other font usages (e.g., normal font and underlining) are indicated by the \f sequences.

1990: groff

Some example groff (GNU troff) code.

Whilst the roff family are not strictly speaking programmable as such, their use of macros means that in practice they are as capable as systems such as TeX (although I don't think that DSLs like LaTeX exist as-such).

Not in the shorter version

1977/1978: TeX

1977/1978 TeX

Presentational with semantic leanings. Programmable. Still in use.

Designed and mostly written by Donald Knuth.

Driven by the need to guarantee accurate typesetting of mathematics.

In serious use of TeX, one starts by defining lots of useful commands - although the TeXbook has many useful ideas one can copy.

In this example, \item is a standard definition, but all of the other commands starting with backslash were defined by my own macros.

1983: LaTeX

1983 LaTeX *Presentational*. Still in use.

Leslie Lamport.

LaTeX is essentially a DSL written in TeX. It's probably still the best known, but certainly not the only one.

I used to write plain TeX, but most people actually use LaTeX, which dates from about 1983/1984. LaTeX is probably still dominant in scientific and mathematical publishing.

This example is from the first edition of the same fanzine - all of the markup is provided for me by LaTeX, so I didn't need to define anything here.

Not in the shorter version

1980: Scribe

1980 Scribe *Presentational*

Described in Brian Reid's 1980 doctoral dissertation at Carnegie Mellon University. Lisp based.

Similar systems still appear to exist.

Note the two representations - the second one being more lisp-like.

1987: TEI

1987 TEI *Semantic*. Still in use today.

"The mission of the Text Encoding Initiative is to develop and maintain a set of high-quality guidelines for the encoding of humanities texts, and to support their use by a wide community of projects, institutions, and individuals"

Some mark up of the start of Swinburne's Sestina, taken from the poetry examples at TEI By Example, showing the working of the ryhming scheme.

`rhyme="ababab"` and then on each line the rhyming word and which part (a, b) of the rhyming scheme it is.

(In the 16x9 version of this slide, I've set these far to the right, to make them more obvious.)

1991: HTML

1991 HTML *Presentational*. Still in use today (although rather altered).

Tim Berners-Lee, at CERN, specified HTML and wrote browser and server software in late 1990. The "HTML Tags" document was first mentioned on the internet in 1991.

HTML 2.0 was published as IETF RFC 1866 in 1995. There was no specification called "HTML 1".

HTML until HTML5 is an SGML document type - an SGML application.

Wikipedia says:

> "The HTML5 syntax is no longer based on SGML despite the similarity of its markup. It has, however, been designed to be backward compatible with common parsing of older versions of HTML. It comes with a new introductory line that looks like an SGML document type declaration, `<!DOCTYPE html>`, which triggers the standards-compliant rendering mode."

1991: Docbook

1991 Docbook *Semantic*. Still in use today.

(Same year as HTML)

"A semantic markup language for technical documentation"

However, I think it is often "semantic" in the same way that LaTeX is "semantic" - often also for presentational purposes (but not *necessarily*).

Example of Docbook 4.3 from http://www.informatik.tu-cottbus.de/~giurca/tutorials/DocBook/index.htm

Before Docbook 5, an SGML language, defined by a DTD

DocBook 5 is an XML language, formally defined by a RELAX NG schema with integrated Schematron rules.

1991: setext

1991 setext *Presentational*. Lightweight.

(Same year as HTML and Docbook)

(This is the beginning of our look at lightweight markup formats)

Ian Feldman, for use in writing the TidBITs electronic newsletter.

Partly a reaction to SGML. Clearly influential on all of the succeeding lightweight markups.

Note: the body text must be indented.

Multi-word italics (`~multiword~italics~`) appears to have been an extension.

Underlining should really mean italics, following typewritten text conventions.

Two dots for comments or special meaning.

Unclear if lists actually were supported. Specification not very clear, specified by examples, not rigorous at all. Really just what he needed for his own purposes.

> (Links look very similar to one of the forms that reStructuredText supports)

1994/1995: wikiwikiweb

1994/1995 wikiwikiweb *Presentational*

The first wiki, invented by Ward Cunningham

Like most wiki formats, specified by example, with no real rigour.

I think that newlines within a paragraph are ignored, but it's hard to tell.

The lack of capability is deliberate, aiming to promote a particular style of discourse:

> "This wiki is quite bare bones, and intentionally so. Less formatting means you have to concentrate on saying things carefully and clearly. Content over form."

Introduced CamelCasedWords as wiki links.

Single quotes - this really distressed me when I first came across it:

- 1 = single quote
- 2 = emphasis
- 3 = bold
- 5 = emphasised bold (2+3)
- 6 are used to stop a CamelCased word from being a WikiLink

Later wiki formats appear not to have understood *why* the design decisions were taken.

1996: StructuredText

1996 StructuredText *Presentational*. Lightweight.

Created by Jim Fulton of Digital Creations (later Zope Foundation) for use in Zope.

Specified by example, somewhat ambiguously.

Clearly influenced by setext.

Significant indentation - good idea in a programming language, not so much when writing plain text.

A heading is a heading because it is followed by a non-heading (!)

Single quotes or doubled backquotes for "inline" text.

Footnotes are fairly simple. Note the use of two dots to introduce the actual footnote.

All block entities must be separated by blank lines.

Note that "o" can be a list delimiter - regarded as a serious ambiguity.

Links are done as:

```
visit the "Python website" :http://www.python.org/.
```

i.e., quoted text followed by a colon and then a URL.

2001/2002: reStructuredText

2001/2002 reStructuredText *Presentational*. Lightweight.

David Goodger had a professional background in SGML.

Original mailing of the idea to the Doc-Sig was in Nov 2000

- Readable is the main aim.
- Output agnostic.
- Well specified, allowing other implementations which behave in the same way.
- Note that < and > are not special - Guido wanted to be able to discuss XML and suchlike without quoting stuff.

Clearly influenced by setext and StructuredText, but with more rigor.

Body text isn't indented (what makes sense for programming languages is irritating for text), but things must line up when appropriate (see the lists).

"o" is not allowed as a list delimiter, as it is too ambiguous.

NB: no way to specify underlined text, which is a Good Thing.

Consciously designed to allow doing certain things but not others - basically, if a document is too complex for reStructuredText, use something like Docbook.

Sphinx was first introduced as a means of using reStructuredText to write the Python documenation, instead of LaTeX.

2002: Asciidoc

2002 Asciidoc *Presentational*. Lightweight.

Stuart Rackham

Aimed specifically as a lightweight way of producing docbook.

Producing docbook means that toolchains exist to produce almost anything else.

The original Asciidoc implementation was written in Python in 2002.

Asciidoctor came out in 2013, and is written in Ruby.

Well specified, allowing other implementations which behave in the same way.

Note the use of underlines to indicate emphasis, a nice look back to typewritten manuscripts.

Paired plus signs for monotyped text.

Use of a + sign to continue a list item into a second paragraph.

Nice (easy to type) way of distinguishing opening and closing quotes.

Footnotes done inline - less readable, but more convenient.

Note that headings can also be delimited with underlining characters, but that doesn't seem to be the normal convention (it's not what the current Asciidoctor documentation introduces, although https://asciidoclive.com still shows that style in its example).

2004: markdown

2004 markdown *Presentation*. Lightweight.

John Gruber, collaborating with Aaron Swartz on the syntax

So nearly a wonderful success.

Yes, I know headings can be underline as well ("setext" style, as it terms it), but I've never seen anyone actually doing that.

- Aimed at producing HTML.

  > From the syntax page: "Markdown's syntax is intended for one purpose: to be used as a format for *writing* for the web." Their emphasis.

- Poorly specified. Ambiguous.
- Allows embedded HTML.
- Most implementations extend it, incompatibly.

Very successful because (the most popular variants) hit a good compromise on the simplicity/capability curve.

Personally, I *think* that markdown would be improved a lot by just removing the ability to embed HTML.

Hopefully CommonMark will improve the situation - for instance, github-flavoured markdown is at least now based on CommonMark.

> The Common Mark spec is at http://spec.commonmark.org/. It is clearly aimed to be a rigourous specification, which is excellent. Note that it calls the underlined heading style "setext headings", which is nice. It still retains the ability to embed HTML in a document, which is not so nice.
>
> The CommonMark specification is also an interesting summary of the problems and incompatibilities of the different implementations, and tries to explain *why* they have made the choices they have made. It is worth reading (although quite long).
>
> However, by the time we've got the rigour of a CommonMark, the complexity of the language seems to me to be at least that of re-StructuredText, without the tidyness of that latter. I think there are many more surprises in how CommonMark "works".

Fin

Since this version of the talk is to be given to Write the Docs, I assume they already know about the Write the Docs website: http://www.writethedocs.org/