

Keywords: logical document structure, representation formats, document types, TEI, LaTeX, DocBook, XML, XHTML.

## Contents

<b>1</b>	<b>Texts, documents and data</b>	<b>1</b>
1.1	The idea of a document . . . . .	3
1.2	Explicit and implicit semantic information . . . . .	3
1.3	Describing or prescribing? . . . . .	6
<b>2</b>	<b>Models for documents</b>	<b>7</b>
2.1	The TeX family . . . . .	7
2.2	Concepts of XML . . . . .	9
2.3	The HTML family . . . . .	12
2.4	The Text Encoding Initiative . . . . .	12
2.5	Docbook . . . . .	17
2.6	The faithful servants . . . . .	19
2.7	The word-processor family . . . . .	20
2.8	Which markup scheme to choose? . . . . .	22
<b>3</b>	<b>Conclusions</b>	<b>22</b>

## Abstract

This chapter considers standards and models for the representation of textual content, both for the purpose of the organisation and presentation of content. It provides an overview of the characteristics and usage of most commonly used document markup formats (OOXML, Open Document Format, TeX/LaTeX, DocBook, Text Encoding Initiative (TEI), XHTML, etc.) with a view on their scope and application. A central theme of the chapter is a clarification of the notion of a *document* in relation to that of *text*. The chapter will overview issues in representing the structure, content and function of documents. Further, it reviews document formats in the light of their support of these representation tasks.

It should be noted that, for convenience, this chapter deals entirely with documents and texts managed in electronic form, as is now almost universally the case. Many of the arguments would apply, however, to older analogue information as well.

## 1 Texts, documents and data

When does text form a document? What are the key characteristics of a structured document, as opposed to structured data or a stream of words? Do documents always consist of words, or can a set of pictures form a document?

There are at least *five* layers of conceptual activity on the spectrum from atomic building blocks to a document.

1. At the bottom level are numbers, letters, image pixels, audio fragments from which we can build all other artefacts (leaving aside for this purpose how those building blocks are stored or a computer).

2. At the next level these blocks are formed into semantically-meaningful artefacts (sentences, pictures, tables, decipherable fragments of audio).
3. The semantic blocks are combined, at some moment in time, to form a storage unit (a file, a database record, a construct in a computer program).
4. One or more storage units presented to a reader or viewer creates a ‘text’, a coherent object which can be described, has metadata, and is created by some rational process.
5. When a ‘text’ has a coherent *purpose*, we may call it a ‘document’.

It is clear that appearance alone, or storage format, or data type, does not dictate to us that we are dealing with a document. If we consider four typical streams of information that we meet on the web daily:

1. an RSS feed of news about rapidly changing current events;
2. a page of images from Flickr resulting from a search for ‘white cats’;
3. information about the current state of share prices;
4. our Inbox of unread email messages.

they are structurally similar to:

1. a set of entries from a scientist’s logbook;
2. a sequence of photos showing the growth of a cell over time;
3. statistical analysis of word usage in the works of Rudyard Kipling;
4. publication of the correspondence between two star-crossed lovers

and may be represented using the same tools and structures on our computer. However, the first set lack the characteristics which would let us describe them as coherent documents, whereas the second set have that extra *purpose* which makes them more than a text.

It should be noted that this taxonomy, distinguishing a ‘text’ from a ‘document’, is not universal; others may distinguish, for example, between a literal transcription of a printed work (a ‘document’), and an edition of that work (a ‘text’). However, for the purpose of this chapter (and other parts of this book), we choose to distinguish early on between the storage of data in digital form (the bits and bytes and other structures), their management on computers (files, etc.), and the intellectually-complete communicating object which is the subject of much of this book.

We conclude this section by asserting that in the field of technical documentation a ‘document’ (and we will omit the ‘emphasis’ in the remainder of this chapter) has at least some of the following characteristics: it has a coherent *purpose*; it has identifiable *authorship* and associated *metadata*; it has a *beginning* and an *end*; and, finally, it has some *structure*. That there may be *several* overlapping structures visible to readers (or even to authors) does not invalidate the necessity of structure.

A document can be created from any useful types of data, combined in a variety of ways, but if it possesses none of these characteristics, it probably falls outside the scope of this chapter. For this book, of course, we may also add the characteristic that the document is *information bearing*, but that may be regarded as an optional extra by some.

## 1.1 The idea of a document

In the previous section, we requested that a document have some *structure*. Conventionally, this will be divided into two parts, the metadata (bibliographical information about the authorship, provenance, classification, etc. of the document) and the main information bearing component. The other chapters in this book will consider in much more detail the meaning of the inner components of text, but here it will be helpful to outline the major components, which in subsequent sections we will compare representations of:

- introductory matter (preface, acknowledgements, summary, etc.);
- main structure, often divided into several ‘chapters’;
- end matter (bibliography or references, notes, etc.).

In the field of scientific documents, we should also be aware of the important category of ‘associated data’, raw material which provides the context or justification of assertions in the text.

It should be clear from the description of components above that the purpose of the structure in a document is to convey *semantic* information within the document. We may recall at this point the debate which lasted through the late 1980s and the 1990s, on the question of ‘what is text’, in which various writers (largely centred around Alan Renear and Steve DeRose) theorized over describing what could be seen in an existing document, and the extent to which it would be reduced to an OHCO (Ordered Hierarchy of Content Objects, codified in SGML, and exemplified by the schemes discussed below). These influential papers (e.g. DeRose et al. (1990), DeRose et al. (1997), Hockey et al. (1999), Renear (1997), and Renear et al. (1996); the philosophers of the digital humanities have returned to the theme many times since) clarified and established for a generation of text encoders the primacy of semantic markup. Later writers refined ideas about ‘text’, establishing that it is a far more fluid affair, especially in the multiple, overlapping, views which can be asserted of it, and showed the limitations of schemes such as XML.

For the discussion in this chapter of technical communication, we are probably safe within the stable OHCO universe.

## 1.2 Explicit and implicit semantic information

One of the major problems in dealing with the instantiation, as opposed to the idea, of a document is the (necessary) use of visual clues to semantic structure in the work as presented to most readers or viewers. While we said in the previous section that a document often has a set of chapters, we only see this by its representation in the page. We deal with a visual set of conventions, derived from centuries of manuscripts and books.

This is well exemplified by a 19th century edition of Euclid, shown in Fig. <sup>1</sup>, which shows a 4-level hierarchy of content:

1. The work: **ELEMENTS OF EUCLID**, using centring, vertical white space, larger font;

---

<sup>1</sup>[http://books.google.com/books?id=ZacAAAAAMAAJ&printsec=frontcover&dq=subject:%22Mathematics%22&hl=en&ei=dGO\\_TNWNdCPNswbL4eWzDQ&sa=X&oi=book\\_result&ct=result&resnum=8&ved=0CD8Q6AEwBw#v=onepage&q&f=false](http://books.google.com/books?id=ZacAAAAAMAAJ&printsec=frontcover&dq=subject:%22Mathematics%22&hl=en&ei=dGO_TNWNdCPNswbL4eWzDQ&sa=X&oi=book_result&ct=result&resnum=8&ved=0CD8Q6AEwBw#v=onepage&q&f=false)

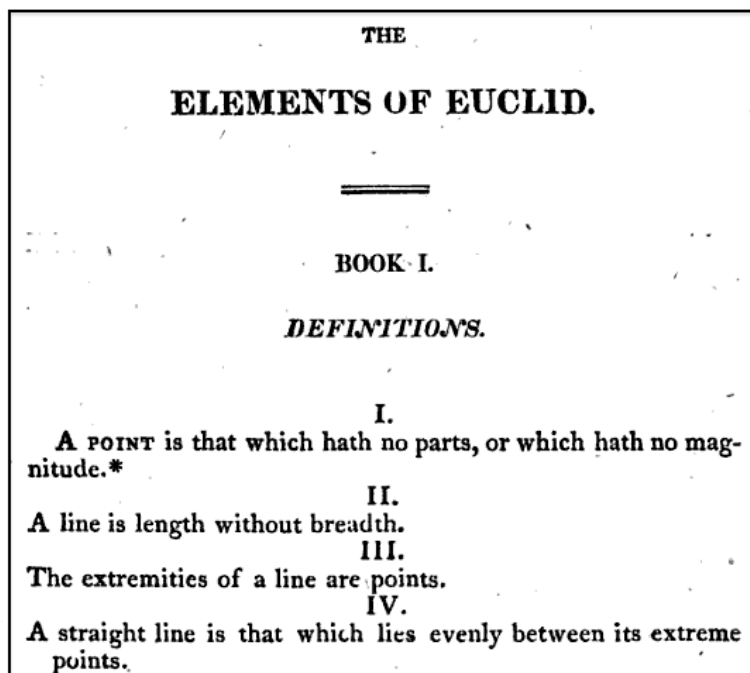


Figure 1: Visual clues to structure in Euclid

2. Book 1: **BOOK I.**, using centring, vertical white space, separating rule, numbering, and capitals;
3. Set of definitions: *DEFINITIONS.*, using centring, italic font and capitals;
4. Definition: **I.**, using centring and numbering.

The use here of whitespace (horizontal and vertical), font changes, colour, and numbering as part of the content, are all relying on several centuries of conventions (since Gutenberg at least); e.g. that an italic heading is usual of lesser importance than one in bold — though the ‘rules’ are not explicitly stated. Contrast the 19th century Euclid with the 3rd century AD Greek papyrus (Fig. <sup>2</sup>) which lacks almost any visual clue (even as to word breaks), and a 10th century manuscript where the scribe has added word breaks but little else.

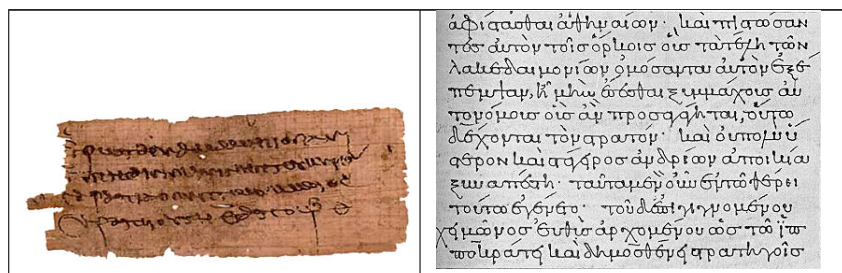


Figure 2: Greek papyrus, Oxyrhynchus Papyri vol. LXII no. 4339 P. Oxy; and 10th century manuscript copy of Thucydides <sup>4</sup>

In the analog form represented above in the papyrus fragments and early book, we have little choice but to use visual clues, because the scribe or typesetter has to use the *same method* to convey both the structure and the content. An alternative in the digital medium is to use *hidden marks* to differentiate true content from content providing meta-information about the semantics of the content. In practical terms, this means using some technique to associate a set of words with a particular intention. How this

We have arrived at the notion that the document is represented by a linear stream of data, of which some components are content ('words') and some are instructions, or markup; and that the markup will instruct a viewing medium on how to present the words to the reader in a manner familiar to them.

We may note that this does not mean a simple 1:1 relationship between markup and visual form. There are, for examples, minor differences like the use of "*semantics*" or «*semantics*» or ,*semantics*' to distinguish a quoted word, high-level cultural differences like reading direction, or (more importantly) the use of a medium like sound, or Braille marks, to convey distinctions in text. Many of the familiar visual techniques (Table ) do come from trying to record patterns of speech (most obviously the use of white space to represent pauses in speech).

Table 2: Commonly-used visual techniques to differentiate text components

Technique	Example
punctuation	this is — or so we believe — the truth; yet we can never know.
white space (vertical or horizontal)	<ul style="list-style-type: none"> <li>• on the one hand</li> <li>• on the other</li> </ul>
change of font	the fixed width font is used to mimic the old typewriter or teletype
change of font style (italic, bold, small-caps, etc.)	the oldest <i>dog</i> in the world, however
change of font size	We may never know what the <small>mouse</small> thought about it
colour	Whatever you do, do not open the bottle marked <b>POISON</b>
page or screen breaks	

What manner of instructions may we insert in the stream of data? There are two distinct models available. Firstly, we can use generalised language to describe the traditional visual effects, in more or less vague terms. Instructions like 'use Helvetica font 8pt bold with 10pt leading' or 'leave 1 cm of white space' are unambiguous, whereas 'emphasise this word', 'center the title on the page' and 'number the item' are more open to different interpretations. They share the characteristic, however, that they are telling a computer programme *what* to do, not *why*. Alternatively, we can use instructions which attempt to explain *why* a group of words is different from its surroundings, without saying how that should be conveyed to the reader. For example, a simple instruction saying 'this is a paragraph' is left open to several effects (Fig. ; text from Rahtz and Rowley (1984)).

Using the latter model, the markup must be accompanied by 'stylesheet'. This term has been used since the start of work on SGML(ISO 1986)) to describe a second (or more) set of instructions which tell a processor how the markup should be formatted; an ISO standard, DSSSL(ISO 1994)) was defined for this in the 1990s, but it was never fully implemented, and the term is now used more commonly in the context of XSL and CSS (see below). A stylesheet clearly provides a more flexible way of working.

Although site 1 appears to be an area excavation and the remainder appear to be trenches of various dimensions the technique employed in all cuttings was the same. The whole of the site was gridded and each cutting was dug in layers.

Features were related to layers as work progressed, finds were recorded on the basis of layer and feature and small finds were additionally plotted three-dimensionally.

---

Although site 1 appears to be an area excavation and the remainder appear to be trenches of various dimensions the technique employed in all cuttings was the same. The whole of the site was gridded and each cutting was dug in layers.

Features were related to layers as work progressed, finds were recorded on the basis of layer and feature and small finds were additionally plotted three-dimensionally.

---

Although site 1 appears to be an area excavation and the remainder appear to be trenches of various dimensions the technique employed in all cuttings was the same. The whole of the site was gridded and each cutting was dug in layers.

Features were related to layers as work progressed, finds were recorded on the basis of layer and feature and small finds were additionally plotted three-dimensionally.

Figure 3: Variations in how paragraphs are displayed

But even when using this model, there remain layers of possible semantic distinctions we can add. Italic ‘emphasis’, for example, is a relatively neutral technique indicating a word or phrase different in some way, and we may often explain more about *why* it is distinct (because it is a name, a title, or a Latin work, for example).

### 1.3 Describing or prescribing?

The combination of abstract markup and a separate interpretative stylesheet is attractive and powerful, and is widely used in modern implementations of technical documentation (although often accompanied by direct visual formatting for when an author wants to do something very specific). For it to be used with any degree of sustainability, there must be agreement on the vocabulary of the markup.

Any markup vocabulary will offer ways of representing common artefacts like text divisions, paragraphs, lists, notes, figures, bibliographies and titles. They may differ, however, on whether they regard these instructions as a framework into which text is placed, or a language to be used to describe what we see on a page. We may distinguish between

**Prescriptive** The text shall have a front matter with title and author, following by one or more sections, an optional set of notes, and a bibliography.

**Descriptive** There are such things as text sections, title statements, and notes, you may combine them as you wish.

The distinction is often made between *new writing* (‘born digital’) which is designed to fit a framework, and *markup of existing material* (‘digitized’) where a third party is trying to describe the intention of an author. It becomes important when we consider schemes to *validate* documents in the next section.

## 2 Models for documents

It is now time to look at the actual methods used to represent markup in technical documentation. There are three dominant models currently popular:

1. the *sui generis* TeX(Knuth 1984) language devised by Donald Knuth, and widely used in its LaTeX(Lamport 1986) variant — we will examine this later;
2. proprietary binary codes in the stream of words, most clearly instantiated by older versions of word-processor formats, such as Wordperfect or Microsoft Word's .doc format — we will not say any more about this, as it is entirely mimicked by the more recent XML-based versions;
3. formats based on XML, such as Text Encoding Initiative, OpenXML(ISO 2008), used current Microsoft Office documents, OpenOffice's Open Document Format(ISO 2006), and most text markup schemes.

The last of these, Extensible Markup Language (XML)(Bray et al. 2008), is a simple, very flexible, text format derived by the Worldwide Web Consortium from SGML(ISO 1986), (Goldfarb 1990) in 1996; it is by some degree the most dominant and accepted format for many varieties of text exchange (even when it is delivered in a compressed package, as Microsoft Office does). For the purposes of this chapter, the language of web pages, HTML(Altheim and McCarron 2001), can be regarded as an instantiation of XML.

### 2.1 The TeX family

The TeX family of markup languages is in a unique position, for several reasons. The language and its implementation<sup>3</sup> are tightly bound together, so that effectively all users run the same software, with very limited interchange possibilities.<sup>4</sup> TeX implements a very rich and powerful typesetting language (especially in the area of mathematics), which can produce high-quality printed or PDF output. However, the input code is *also* a programming language, which allows for high-level markup languages to be developed on top of it.

The best known language written using TeX is LaTeX(Lamport 1986), which defines a notation to address the same high-level constructs as we listed in the previous section. The expressiveness of this is extended by a very well-developed community of add-on packages, allowing the author to describe everything from music to chess games to complex bibliographies all in the same system, and receive reliable output. LaTeX also has important external tools to assist with smart editing, bibliography formatting and citation management (BibTeX), and index term management (makeindex). Combined with the sophistication of the underlying math engine, this has meant LaTeX is adopted very widely for preprints, theses, and scientific book typesetting.

A simple LaTeX document is listed in Fig. .

---

<sup>3</sup>The current TeX software, under the close control of its author Donald Knuth, was released in 1982(Knuth 1984) and has changed very little since that time. Despite some considerable additions of capability in the last decade, most notably pdfTeX for making PDF output, and XeTeX for processing Unicode, the adherence to the input language and backward compatibility is almost absolute. TeX is one of the oldest and cleanest examples of free software.

<sup>4</sup>Conversions from TeX to other formats are distinctly non-trivial, because of the nature of its input language, and TeX itself supports no import from other formats, though it can be programmed to read XML. Conversions to TeX from XML formats are commonplace and easy to write.

```

\documentclass[11pt,twoside]{article}
\usepackage{times,hyperref}
\date{January 17, 1999}
\title{Simulation of Energy Loss Straggling}
\author{Maria Physicist}
\begin{document}
\maketitle
\section{Introduction}\label{intro}

\par Due to the statistical nature of ionisation energy loss,
large fluctuations can occur in the amount of energy deposited
by a particle traversing an absorber element. Continuous
processes such as multiple scattering and energy loss play a
relevant role in the longitudinal and lateral development of
electromagnetic and \emph{hadronic showers}, and in the case
of sampling calorimeters the measured resolution can be
significantly affected by such fluctuations in their
active layers.

\par An example formula
\begin{equation}
\xi / I_0
\end{equation}
\par The work at \href{http://public.web.cern.ch/public/}{CERN}
is likely to be of interest to readers of this paper.
\end{document}

```

Figure 4: Simple document in LaTeX



LaTeX is a tightly-coupled system. The same language is used to describe both what is required ('centre this paragraph and set it in smaller than the surrounding paragraphs') and to program interaction with the output page ('if there is less than 3 inches left on the page, set the text slightly smaller'). This built-in algorithmic rule-based working is quite distinct from either the separate stylesheet approach of the XML languages we shall discuss in the next sections, or with the working of word-processor systems like Microsoft Word which assume human *ad hoc* interaction.

The problems with LaTeX are, however, serious. Its extensibility and macro-programming ability make it very hard to do meaningful validation of the structure, or process it with other generic tools. *[Note: It is possible, using a structured editor, to write entirely 'vanilla' LaTeX which is amenable to structured processing, but it is not easy to enforce this.]* The relative difficulty of generating high-quality web pages, or summarising semantic information eg as RDF, seem to place LaTeX at one end of a write-only cul-de-sac.

## 2.2 Concepts of XML

XML provides a general model of structured data represented as strings of text. It has a simple convention that all instructions are enclosed in < and > characters, and has two important characteristics: firstly, the set of instructions is not prescribed, allowing anyone to define a vocabulary; and secondly, there is an added concept of a schema, or set of rules, for a vocabulary, which allows a document to be checked or *validated*.

An XML document represents a tree. It has a single root and many nodes. Each node can be a subtree, a single *element* (possibly bearing some *attributes*) or a string of *character data*. There are also *comments*, *processing instructions*, *entity references* and *marked sections*, but these are obscure and rarely encountered. The main tool is the *element* (container) and its *attributes*; the basic syntax is shown in the following:

```
<?xml version="1.0" encoding="utf-8"?>
<rootElement>
  <elementName attributeName="attributeValue">
    elementContent
  </elementName>
  <!-- this is a comment -->
</rootElement>
```

Thus an XML document is encoded as a linear string of characters. It begins with a special instruction (the XML declaration) which specifies that this *is* an XML document, and which version of the XML standard it follows; it may also specify a different character encoding for the document. Elements are then marked by start- and end-tags using < and > around a name. Comments are delimited by <!-- and --> and attribute name/value pairs are supplied on the start-tag and may be given in any order.

The elements can be in different *name spaces*, to allow for the same name being found in different vocabularies, by using a special attribute `xmlns`. The following example shows how math markup is differentiated by its namespace:

```

<p>This is some text, and now
some math: <m:math>
  <m:msub>
    <m:mrow>
      <m:mi>p</m:mi>
    </m:mrow>
    <m:mrow>
      <m:mi mathvariant="normal">T</m:mi>
    </m:mrow>
  </m:msub>
</m:math>
</p>

```

Which elements are allowed (<p>, <math>, <msub>, <mi>, <mrow> in the previous example), and in what order or combination, is dictated by a separate *schema* document, which can be written in a variety of notations. Checking the document against the schema is called *validation*. The most common languages are Document Type Definition (DTD) language of XML, W3C Schema(Thompson et al. 2004), and RELAX NG(ISO 2003); they let the vocabulary designer say:

1. What the root element may be;
2. What other elements, or character data, are allowed within a given element;
3. What attributes are allowed, and whether they are optional or mandatory;
4. What sort of data is allowed inside attributes or elements (ie numbers, dates, URIs, choices from a fixed list).

The intention of this is very similar to that of a database schema, determining what sort of data can be put into a container, and whether it is mandatory to do so. Fig. shows a very simple schema, written in RELAXNG, and an XML document valid against it.

Each of the common schemas for technical documentation (OpenOffice XML, Open Document Format, Docbook, HTML, and TEI) provides a similar set of constructs and rules. They each describe

- Metadata about document
- Title and author
- Headings, indicating a hierarchy of sections
- Paragraphs
- Lists
- Tables
- Figures
- Hyperlinks
- Cross-referencing

and a wide variety of inline emphasis constructs. They differ greatly, however, in the combinations allow, the way in which arbitrary formatting or effects is supported, and the provision of specialised vocabulary for particular domains.

```

<rng:grammar datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <rng:define name="people">
    <rng:element name="people">
      <rng:oneOrMore>
        <rng:ref name="person"/>
      </rng:oneOrMore>
    </rng:element>
  </rng:define>
  <rng:define name="person">
    <rng:element name="person">
      <rng:attribute name="birth">
        <rng:data type="gYear"/>
      </rng:attribute>
      <rng:optional>
        <rng:attribute name="death">
          <rng:data type="gYear"/>
        </rng:attribute>
      </rng:optional>
      <rng:text/>
    </rng:element>
  </rng:define>
  <rng:start>
    <rng:ref name="people"/>
  </rng:start>
</rng:grammar>

```

```

<people>
  <person birth="1832" death="1888">Alcott, Louisa May</person>
  <person birth="1775" death="1817">Austen, Jane</person>
  <person birth="1861" death="1891">Balestier, Wolcott</person>
  <person birth="1757" death="1827">Blake, William</person>
  <person birth="1848" death="1895">Boyesen, Hjalmar
Hjorth</person>
  <person birth="1820" death="1849">Brontë, Anne</person>
  <person birth="1818" death="1848">Brontë, Emily</person>
  <person birth="1857" death="1924">Conrad, Joseph</person>
  <person birth="1883" death="1923">Hašek, Jaroslav</person>
  <person birth="1865" death="1936">Kipling, Rudyard</person>
  <person birth="1931">Le Carré, John</person>
</people>

```

Figure 5: Minimal schema and XML document

## 2.3 The HTML family

By far the most widely used (and most abused) markup language is HTML (Altheim and McCarron 2001). Although influenced by SGML in its first years, it was only in the mid 1990s formalised as an SGML profile, and has seldom been fully implemented as such by processing tools (i.e. web browsers). An explicit XML-compliant version, XHTML, was completed in 2000, but does not have full acceptance 10 years later; that is now being supplanted by a second branch, HTML5 (Hickson 2010), which extends the language at the same time, but is not expected to have total adoption for some years.

A simple HTML document is listed in Fig. , showing the same document as LaTeX in the section above. However, HTML is normally used much more as *container* for a structured document, wrapping it a layer of presentation, navigation and dynamic features. It has two helper technologies which make it hard to treat it on a par with other schemes:

- CSS (Cascading Stylesheet language) (Çelik et al. 2009): this provides a way to attach visual properties (fonts, spacing, colours, etc.) to HTML elements, keyed by means of the *@class* attribute visible on the `<div>` elements in Fig. ;
- Javascript (ISO 2009): all almost web browsers implement this scripting language for manipulating the page, pulling in new data, and performing actions.

The use of Javascript in particular can make HTML a tightly-coupled language like LaTeX, where the page can change itself dynamically inside the browser. Crucially, however, HTML can also be used in other contents, and with other software, as a much more general purpose language. An example of its reuse is as the underlying formatting language of the ePub electronic book specification. (IDPF 2010)

The most powerful technique in HTML is its use of two very generic nestable container elements, `<span>` and `<div>`, which can be used with the *@class* attribute to model almost anything. Additional generic attributes added by the Microdata (Hickson 2011) or RDFa (Pemberton et al. 2008) W3C initiatives also make it possible to put an extra layer of semantic markup over HTML.

New elements in HTML5 allow for an even closer relationship with the implementation, in that use of conforming browsers will not just allow standardised video embedding, but also access to environmental information as the location supplied by a computer's GPS. This pushes HTML, and its companion Javascript, closer to a functional programming language, and away from a document representation language.

## 2.4 The Text Encoding Initiative

The Text Encoding Initiative (TEI) (TEI Consortium 2010) scheme was developed during the 1990s, and then again after 2000, now at a 5th release, to create a very rich schema which can reasonably claim to have the most wide-ranging and detailed expressivity of all the markup languages. The following table shows the choice of elements available to a writer inside a paragraph, with the TEI offering a staggering 182 possibilities in its fullest version.

Number of different elements permitted inside paragraph	Markup language
182	TEI (all)
146	Docbook

TEI was designed as *descriptive* scheme for digitisation of existing material from across the literary and linguistic spectrum, emphasising the use of markup for analysis rather than display.

A simple TEI document is listed in Fig. . Notice the use of a container <div>, with a child <head> rather than the simple heading approach of HTML's <h1> and <h2>. In contrast, consider the short fragment of speech 'Enough enough, do you want to go buy some Polo?', marked up below with linguistic part of speech information using TEI elements:

```
<u who="#PS05C">
  <s n="1194">
    <w lemma="enough" type="AV0">Enough </w>
    <w lemma="enough" type="AV0">enough</w>
    <c type="PUN">,< /c>
    <w lemma="do" type="VDB">do </w>
    <w lemma="you" type="PNP">you </w>
    <w lemma="want" type="VVI">want </w>
    <w lemma="to" type="TO0">to </w>
    <w lemma="go" type="VVI">go </w>
    <w lemma="buy" type="VVB">buy </w>
    <w lemma="some" type="DT0">some </w>
    <w lemma="polo" type="NN1">Polo</w>
    <c type="PUN">?</c>
  </s>
</u>
```

TEI offers so much choice for two reasons: firstly, it is a modular scheme covering a broad range of descriptive and analytic markup (Table lists the currently available modules), and secondly because of its descriptive nature. Where a prescriptive scheme might dictate that lists are not permitted inside paragraphs (HTML is an example), the TEI must allows both inside and outside paragraphs, because it cannot dictate where such things may be found in existing text.

Amongst TEI's modules there are several which are more akin to database schemas than traditional text. For example, the following fragment is a description of medieval manuscript, with a bewildering variety of metadata in attributes, formal 'fields' (eg <rubric>) and use of inline transcriptional notes, eg on the text of <explicit>.

```
<msItem n="c">
  <locus from="2r9" to="2r19">2r:9-2r:19</locus>
  <title type="uniform">Fra paradiso</title>
  <rubric>Fra paradiso</rubric><incipit>Sva er sagt at paradis er hi<expant>n</expant>n oest
  lutr pessarrar veraldar.</incipit><explicit>oc veria bio<supplied reason="illegible">r</s
  hitar at me<expant>n</expant>n skili eigi pangat komast.</explicit></msItem>
```

```

<xhtml:html lang="en">
  <xhtml:head>
    <xhtml:meta http-equiv="Content-Type"
      content="text/html; charset=UTF-8"/>
    <xhtml:title>Simulation of Energy Loss Straggling</xhtml:title>
    <xhtml:meta name="author"
      content="Maria Physicist"/>
    <xhtml:meta name="DC.Title"
      content="Simulation of Energy Loss Straggling"/>
  </xhtml:head>
  <xhtml:body>
    <xhtml:div class="header">
      <xhtml:h1 class="title">Simulation of Energy Loss
Straggling</xhtml:h1>
      <xhtml:h2 class="author">Maria Physicist</xhtml:h2>
    </xhtml:div>
    <xhtml:div class="div1" id="intro">
      <xhtml:h2>
        <xhtml:span class="head">Introduction</xhtml:span>
      </xhtml:h2>
      <xhtml:p>Due to the statistical nature of ionisation energy
loss,
        large fluctuations can occur in the amount of energy
        deposited by a particle traversing an absorber element.
        Continuous processes such as multiple scattering and
energy
        loss play a relevant role in the longitudinal and
lateral
        development of electromagnetic and <xhtml:span class="term">hadronic
        showers</xhtml:span>, and in the case of
        sampling calorimeters the measured resolution can be
        significantly affected by such fluctuations in their
active
        layers. </xhtml:p>
      <xhtml:p>The work at <xhtml:a href="http://public.web.cern.ch/public/">CERN</xhtml:a>
is
        likely to be of interest to readers of this
paper.</xhtml:p>
    </xhtml:div>
    <xhtml:div class="footer">
      <xhtml:address> Maria Physicist.
        Date: January 17, 1999
      </xhtml:address>
    </xhtml:div>
  </xhtml:body>
</xhtml:html>

```

Figure 6: Simple document in HTML

```

<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>Simulation of Energy Loss Straggling</title>
        <author>Maria Physicist</author>
      </titleStmt>
      <editionStmt>
        <edition>
          <date>January 17, 1999</date>
        </edition>
      </editionStmt>
      <publicationStmt>
        <p>unpublished</p>
      </publicationStmt>
      <sourceDesc>
        <p>part of a sample document used at CERN for teaching
LaTeX</p>
      </sourceDesc>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <div xml:id="intro">
        <head>Introduction</head>
        <p>Due to the statistical nature of ionisation energy loss,
          large fluctuations can occur in the amount of energy
          deposited by a particle traversing an absorber
element.
          Continuous processes such as multiple scattering and
energy
          loss play a relevant role in the longitudinal and
lateral
          development of electromagnetic and <term>hadronic
          showers</term>, and in the case of sampling
calorimeters
          the measured resolution can be significantly affected
by
          such fluctuations in their active layers. </p>
        <p>An example formula:
        <formula notation="MathML">
          <m:math>
            <m:mi> $\xi$ </m:mi>
            <m:mo>/</m:mo>
            <m:msub>
              <m:mi>I</m:mi>
              <m:mrow>
                <m:mn>0</m:mn>
              </m:mrow>
            </m:msub>
          </m:math>
        </formula>
        </p>
        <p>The work at <ref target="http://public.web.cern.ch/public/">CERN</ref>
          is likely to be of interest to readers of this
paper.</p>
      </div>
    </body>
  </text>
</TEI>

```

Figure 7: Simple document in TEI

Table 4: Modules in Text Encoding Initiative scheme

<i>Module name in TEI</i>	<i>Coverage</i>
analysis	Simple analytic mechanisms
certainty	Certainty and responsibility
core	Elements available in all TEI documents
corpus	Language corpora
dictionaries	Dictionaries
drama	Performance texts
figures	Tables, formulae, and graphics
gaiji	Representation of non-standard characters and glyphs
header	The TEI Header
iso-fs	Feature structures
linking	Linking, segmentation, and alignment
msdescription	Manuscript description
namesdates	Names, dates, people, and places
nets	Graphs, networks, and trees
spoken	Transcriptions of speech
tagdocs	Documentation elements
tei	The TEI infrastructure
textcrit	Critical apparatus
textstructure	Default text structure
transcr	Representation of primary sources
verse	Verse

One of the TEI's key design features is its assumption that users will always choose from a subset of the elements available (the TEI Consortium provides tools to make this easy, eg Roma(TEI 2010)), and thus never face the choice of 182 elements. As the table earlier showed, a typical TEI profile ('Lite') only offers 62 choices inside a paragraph — though this can still be offputtingly large.

From the point of view of technical documentation interchange, the TEI is less than ideal because it uses very general mechanisms typically qualified by unconstrained *@type* attributes. For example, where HTML models a numbered list as

```
<ul>
  <li>first item</li>
  <li>second item</li>
</ul>
```

the TEI will use

```
<list type="ordered">
  <item>first item</item>
  <item>second item</item>
</list>
```

The freedom allowed here for other values than 'ordered' for *@type* can be confusing to the consumer of a TEI text.



In practice, users of the TEI group into *communities of practice*, adopting more strictly controlled subsets of the specification for a particular domain or workflow. For example, scholars using TEI to encode detailed descriptions of medieval manuscripts may use the subset defined by ENRICH (<http://enrich.manuscriptorium.com/>), while large-scale, but light touch, library digitization projects may use the *TEI in Libraries: Guidelines for Best Practices* ([http://wiki.tei-c.org/index.php/TEI\\_in\\_Libraries:\\_Guidelines\\_for\\_Best\\_Practices\\_Working\\_Group](http://wiki.tei-c.org/index.php/TEI_in_Libraries:_Guidelines_for_Best_Practices_Working_Group)). Interoperability between a manuscript description and a library digital edition of *A Christmas Carol* is rarely a high priority, so using very distinct subsets of TEI is not problematic. The two documents do, however, share a common ontology (regarding the TEI simply as a vocabulary), so future use of the two streams of data is considerably easier than if they had used entirely different schemata.

## 2.5 Docbook

Docbook (Walsh 2010), developed since the 1990s, and now in its 5th release, is the best-known and best-supported of the technical writing schemas, with support from many tool vendors. In contrast to the TEI, it is highly prescriptive. It is squarely aimed at new writing, and supporting features for documenting IT systems (it descends from the requirements of the technical publisher O'Reilly. The simple example is shown in Fig. , making no use of Docbook's specialised elements. For example (list taken from the documentation) Docbook offers five distinct ways of including examples of computer programs:

**<literallayout>** A `<literallayout>` does not have any semantic association beyond the preservation of whitespace and line breaks. In particular, while `programlisting` and `screen` are frequently presented in a fixed-width font, a change of fonts is not ordinarily implied by `<literallayout>`.

**<programlisting> and <programlistingco>** The `<programlisting>` and `<programlistingco>` elements are verbatim environments, usually presented in Courier or some other fixed-width font, for program sources, code fragments, and similar listings.

**<screen> and <screenco>** The `<screen>` and `<screenco>` elements are verbatim or literal environments for text screen captures, other fragments of an ASCII display, and similar things.

**screenshot** `<screenshot>` is actually a wrapper for a `<mediaobject>` intended for screenshots of a GUI, for example.

**synopsis** A synopsis is a verbatim environment for command and function synopses.

In even more detail, there are 14 elements for describing aspects of a user interface (`<accel>`, `<guibutton>`, `<guiicon>`, `<guilabel>`, `<guimenu>`, `<guimenuitem>`, `<guisubmenu>`, `<keycap>`, `<keycode>`, `<keycombo>`, `<keysym>`, `<menuchoice>`, `<mousebutton>`, and `<shortcut>`) and 17 more for specifying a computer language. Like the TEI, however, Docbook offers general purpose elements and attributes which can allow authors to ignore the semantic elements

Docbook gains considerably from solid support in tools for rendering into different formats (HTML, PDF, etc.) and from its well-documented and detailed syntax for computer documentation. Like the TEI, it has a rich and powerful section of elements dedicated to metadata about the document.

```

<dbk:article version="5.0"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <dbk:info>
    <dbk:title>Simulation of Energy Loss Straggling</dbk:title>
    <dbk:author>
      <dbk:personname>Maria Physicist</dbk:personname>
      <dbk:address>
        <dbk:city>Geneva</dbk:city>
        <dbk:postcode>CH-1211 </dbk:postcode>
        <dbk:country>Switzerland</dbk:country></dbk:address>
        <dbk:email>maria.physicist@cern.ch</dbk:email></dbk:author>
      <dbk:pubdate>January 17th, 1999</dbk:pubdate></dbk:info>
    <dbk:sect1>
      <dbk:title>Introduction</dbk:title>
      <dbk:para>Due to the statistical nature of ionisation energy
loss,
      large fluctuations can occur in the amount of energy
deposited by
      a particle traversing an absorber element. Continuous
processes
      such as multiple scattering and energy loss play a
relevant role
      in the longitudinal and lateral development of
electromagnetic and
      <dbk:firstterm>hadronic showers</dbk:firstterm>, and in the
case of
      sampling calorimeters the measured resolution can be
significantly
      affected by such fluctuations in their active
layers.</dbk:para>
      <dbk:para>An example formula:
      <dbk:equation>
        <m:math>
          <m:mi> $\xi$ </m:mi>
          <m:mo>/</m:mo>
          <m:msub>
            <m:mi>I</m:mi>
            <m:mrow>
              <m:mn>0</m:mn>
            </m:mrow>
          </m:msub>
        </m:math>
      </dbk:equation></dbk:para>
      <dbk:para>The work at <dbk:link xlink:href="http://public.web.cern.ch/public/">CERN</>
is
      likely to be of interest to readers of this
paper.</dbk:para></dbk:sect1></dbk:article>

```

Figure 8: Simple document in Docbook

## 2.6 The faithful servants

The schemas we have described above all cover the broad aspects of describing the semantics of a document, specifying metadata, and (in different areas) delve into domain-specific markup. However, there remain four specialist areas in which (wisely) none of the XML-based schemes venture. These are catered for by XML standards of their own, largely from the Worldwide Web Consortium, which can be included within any document by virtue of *namespaces*. These specialist schemes are for:

**Mathematics** The MathML(Poppelier et al. 2003) language provides comprehensive facilities for expressing maths in XML markup, both in a conventional presentation way, and in a less familiar but more powerful semantic fashion.

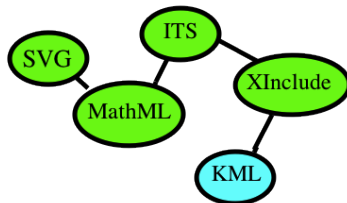
**Vector graphics** Scaleable Vector Graphics (SVG)(Dahlström et al. 2010) defines a language for pictures, such as graphs and plots, primarily targetting the web, where it also provides for user interaction.

**Internationalization** The Internationalization Tag Set (ITS)(Sasaki and Lieske 2007) provides facilities for marking up multi-lingual documents, eg specifying writing direction, and for translation tools, indicating where translation should occur.

**Transclusion** The small but necessary XInclude(Marsh et al. 2006) standard provides markup for one document pulling in bits of another.

**Describing geographical maps** The KML(Google 2010) and GML(ISO 2007) languages offer descriptions of mapping at a higher level than SVG; the former is in very widespread use through Google Earth and Google Maps.

We have already seen an example of MathML in our sample documents earlier, and those conveniently illustrate how two document schemas can work together. Using SVG is rather similar. Consider this picture:



It look like this in SVG XML:

```
<ellipse id="svg_10"
  transform="rotate(-0.516164243221283 305.51266479492284,100.40528869628734)
  " filter="url(#svg_7_blur)" ry="31.56728"
  rx="60.44972" cy="100.40529" cx="305.51266"
  stroke-width="5" stroke="#000000" fill="#00ff00"/>
```

The existence of the servant schemas is important, because they demonstrate a possibility of document interchange and re-use beyond the relatively simple word-based systems. However, we should consider that the MathML and SVG languages are well over a decade old, but are not yet reliably supported by all the components of the modern web, and be a little cautious.

## 2.7 The word-processor family

The word-processor formats of the common office suites (Microsoft Office 2007 onwards, and Open Office(OpenOffice 2010)) store their documents in XML markup (albeit as a ZIP archive containing a number of files). They each have a model which combines a very light structural model (fundamentally little more than a set of blocks on the page, with inline sequences inside the blocks) with a very rich set of visual properties, addressing combinations with named *styles*. This is potentially very powerful, when used in a disciplined way, but the set of styles can be very confusing, ranging from the typical structural functions (Fig. (1)), through purely presentational (Fig. (2)), to locally-defined domain specific (Fig. (3)). The software also allows the user to override any of the formatting *ad hoc*, which can create havoc for interchange.

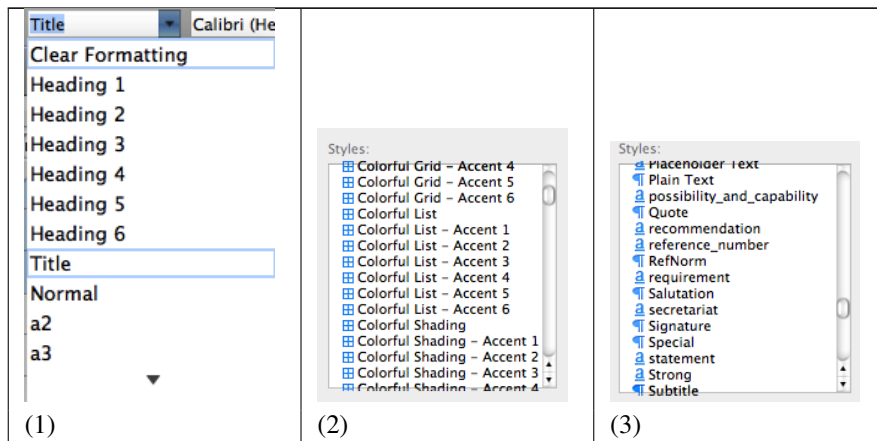


Figure 9: Style listing in Microsoft Word

The underlying XML is relatively clean, since all of the styling information is abstracted away from the normal stream. The representation of a simple paragraph earlier in this chapter looks like this in in Open XML(ISO 2008):

```
<w:p
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:pStyle w:val="TextPara"/>
  <w:r>
    <w:rPr/>
    <w:t xml:space="preserve">An XML document represents a (kind of) tree. It
has a single root and many nodes Each node can be a subtree, a single
</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:b/>
    </w:rPr>
    <w:t xml:space="preserve">element</w:t>
  </w:r>
```

```

<w:r>
  <w:rPr/>
  <w:t xml:space="preserve"> (possibly bearing some
</w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:b/>
  </w:rPr>
  <w:t xml:space="preserve">attributes</w:t>
</w:r>
<w:r>
  <w:rPr/>
  <w:t xml:space="preserve">) or a string of
</w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:b/>
  </w:rPr>
  <w:t xml:space="preserve">character data.</w:t>
</w:r>
</w:p>

```

and like this in Open Document Format(ISO 2006) (ODF):

```

<text:p text:style-name="Text body"
xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0">An
XML document represents a (kind of)
tree. It has a single root and many nodes Each node can be a
subtree,
a single <text:span text:style-name="Emphasis">element</text:span>
(possibly bearing some <text:span text:style-name="Emphasis">attributes</text:span>)
or a string of
<text:span text:style-name="Emphasis">character
data.</text:span>
</text:p>

```

The difference between the formats is that OpenXML keeps *all* text within a spanning container, whereas Open Document Format allows ‘mixed content’ inside `<text:p>`, a mixture of character data and `<text:span>` elements. OpenXML allows for both a style reference (`<pStyle val="TextPara"/>`) and a local override (`<rPr><b/></rPr>` says that the text should be in bold), where ODF has simply style names.

OpenXML makes especially heavy use of namespaces, defining 12 separate vocabularies for different parts of the document, which it represents using at least 20 separate files. The apparent simplicity of the markup example above masks considerably detailed specification of visual formatting, the details of tables, and hyperlinking.

The word-processor schemes are tightly coupled to presentation and the software. Separating out all the visual formatting to named style components, they operate within a single level of nesting, and only the style name provides any attempt at semantic

information. There is no possibility of rules to say where styles can occur or nest, although such control can be programmed within the software itself.

## 2.8 Which markup scheme to choose?

It is unlikely that any of the document representation schemes described above is a universal panacea suited to everyone. It is clear that their characteristics are dictated by very different requirements, and a choice will depend on an assessment of tool support, validation needs, domain coverage, output needs, and peer-group involvement. The use of specialised vocabularies for math, graphics, etc. seems unlikely to be controversial, as the only alternative is to provide non-scaling pictures.

Whichever scheme is adopted to manage the structured documents we defined at the start of this chapter, it is inevitable that there will be *abuse* — the LaTeX users who makes section headings by setting a large bold sans-serif font, the HTML author who uses tables to lay out pages, the TEI encoder who uses `<1>` (designed for lines of verse) to force line breaks, the Docbook writer who uses the `@style` attribute to convey all distinctions, and the Word practitioner who puts in empty paragraphs to make more vertical space.

## 3 Conclusions

We started by describing the characteristics of a ‘document’, and we have spent most of this chapter looking at a variety of document markup schemes and how they manage the relationship between that abstract document and its visual representation. There are three main reasons for making such a distinction explicit.

The first is the desire to be able to *change* the visual representation for different situations or media, including non-visual representation to provide access for a wider range of readers.

The second is the ability to *check* the structure of documents, and ensure that they contain what we think they do.

The third is the need to *analyse* documents, and either extract data from them, or enhance them with eg links to other documents and resources.

The first two of these claims, eloquently argued in a classic paper of 1987 by Coombs, Renear and de Rose (Coombs et al. 1987) (and in many other publications since), may be open to the criticism that the writing and publishing industry has *not*, generally speaking, adopted the idea of separating markup from content. Phrases like ‘The most dramatic sign of change is the overwhelming promotion of SGML’ from 1990 (DeRose et al. (1990), p.16) raise a smile 20 years later. Most writing remains tied to a particular medium, and uses *ad hoc* visual formatting. Much use is made of design features which make documents attractive, but have no semantic corollary.

It is more convincing, however, to argue that the separate representation of content, structure and layout in a document is of far more importance than just layout, because it provides the possibility of identifying, extracting and sharing *data* from documents, which brings us back to the start of the chapter. A document is more than simply a stream of data, but that does not mean it does not contain data within it. Whether that be personal names, dates, formulae, geographical locations, addresses, financial data or references, the structured document is the building block of the modern information flow.

## References

- Murray Altheim and Shane McCarron. XHTML 1.1 - Module-based XHTML. Recommendation, W3C, 2001. URL <http://www.w3.org/TR/2001/REC-xhtml11-20010531>. 2010-10-25.
- Tim Bray, Jean Paoli, Eve Maler, François Yergeau, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 (Fifth series). Recommendation, W3C, 2008. URL <http://www.w3.org/TR/2008/REC-xml-20081126/>. 2010-10-25.
- James H. Coombes, Allen Renear, and Steven J. DeRose. Markup Systems and The Future of Scholarly Text Processing. *Communications of the Association for Computing Machinery*, 30, 1987.
- Erik Dahlström, Jon Ferraiolo, Fujisawa Jun, Anthony Grasso, Dean Jackson, Chris Lilley, Cameron McCormack, Doug Schepers, Jonathan Watt, and Patrick Dengler. Scalable Vector Graphics (SVG) 1.1 (Second Edition). Working Draft, W3C, 2010. URL <http://www.w3.org/TR/2010/WD-SVG11-20100622/>. 2010-10-25.
- Steven J. DeRose, David Durand, Elli Mylonas, and Allen Renear. What is Text, Really? *Journal of Computing in Higher Education*, 1:3–26, 1990.
- Steven J. DeRose, David G. Durand, Elli Mylonas, and Allen H. Renear. Further Context for “What is Text, Really?”. *The Journal of Computer Documentation*, 21: 40–44, 1997.
- Charles F. Goldfarb. *The SGML Handbook*. Oxford, Clarendon Press, 1990.
- Google. KML (Keyhole Markup Language). Technical report, Google, 2010. URL <http://code.google.com/apis/kml/documentation/>. 2010-10-25.
- Ian Hickson. HTML5: A vocabulary and associated APIs for HTML and XHTML. Working Draft, W3C, 2010. URL <http://www.w3.org/TR/2010/WD-html5-20101019/>.
- Ian Hickson. HTML Microdata. Recommendation, W3C, 2011. URL <http://www.w3.org/TR/microdata/>. 2011-09-14.
- Susan Hockey, Allen Renear, and Jerome J. McGann. Panel: What is Text? A Debate on the Philosophical and Epistemological Nature of Text in the Light of Humanities Computing Research. Technical report, Association of Literary and Linguistic Computing, 1999. URL <http://www.iath.virginia.edu/ach-allc.99/proceedings/hockey-renear2.html>.
- IDPF. ePub specification. Technical report, IDPF, 2010. URL <http://www.idpf.org/specs.htm>. 2010-10-25.
- ISO. *Standard Generalized Markup Language (ISO 8879:1986 SGML)*. International Organization for Standardization, Geneva, 1986.
- ISO. *Information processing — Text and office systems — Document Style Semantics and Specification Language (DSSSL), ISO/IEC DIS 10179.2*. International Organization for Standardization, Geneva, 1994.

- ISO. *ISO/IEC 19757-2:Ampl Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG – Amendment 1: Compact Syntax. ISO version of the RELAX NG Compact Syntax*. International Organization for Standardization, Geneva, 2003.
- ISO. *ISO/IEC 26300:2006 Information technology – Open Document Format for Office Applications (OpenDocument) v1.0*. International Organization for Standardization, Geneva, 2006.
- ISO. *ISO 19136:2007 Geographic information – Geography Markup Language (GML)*. International Organization for Standardization, Geneva, 2007.
- ISO. *ISO/IEC 29500-1:2008 Information technology – Document description and processing languages – Office Open XML File Format – Part 1: Fundamentals and Markup Language Reference*. International Organization for Standardization, Geneva, 2008.
- ISO. *ISO/IEC 16262:2002 Information technology - ECMAScript language specification*. International Organization for Standardization, Geneva, 2009.
- Donald E. Knuth. *The TeXbook*. Addison-Wesley, Reading, Massachusetts, 1984.
- Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 1986.
- Jonathan Marsh, David Orchard, and Daniel Veillard. XML Inclusions (XInclude) Version 1.0 (Second Edition). Recommendation, W3C, 2006. URL <http://www.w3.org/TR/2006/REC-xinclude-20061115/>. 2010-10-25.
- OpenOffice. OpenOffice.org. Technical report, Oracle, 2010. URL <http://www.openoffice.org/>. 2010-10-25.
- Steven Pemberton, Ben Adida, Shane McCarron, and Mark Birbeck. RDFa in XHTML: Syntax and Processing. A collection of attributes and processing rules for extending XHTML to support RDF. Recommendation, W3C, 2008. URL <http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014>. 2010-10-25.
- Nico Poppelier, Robert Miner, Patrick Ion, and David Carlisle. Mathematical Markup Language (MathML) Version 2.0 (Second Edition). Recommendation, W3C, 2003. URL <http://www.w3.org/TR/2003/REC-MathML2-20031021/>. 2010-10-25.
- Sebastian Rahtz and Trevor Rowley. *Middleton Stoney. Excavation and Survey in a North Oxfordshire Parish 1970–1982*. Oxford University Department for External Studies, 1984.
- Allen Renear. Out of Praxis: Three (Meta)Theories of Textuality. In Kathryn Sutherland, editor, *Electronic Text. Investigations in Method and Theory*, pages 107–126. Clarendon Press, Oxford, 1997.
- Allen Renear, Elli Mylonas, and David Durand. Refining our Notion of What Text Really Is: The Problem of Overlapping Hierarchies. In Nancy Ide and Susan Hockey, editors, *Research in Humanities Computing*, pages 263–280. Oxford University Press, Oxford, 1996.



- Felix Sasaki and Christian Lieske. Internationalization Tag Set (ITS) Version 1.0. Recommendation, W3C, 2007. URL <http://www.w3.org/TR/2007/REC-its-20070403/>. 2010-10-25.
- TEI. Roma: generating validators for the TEI. Technical report, Text Encoding Initiative Consortium, 2010. URL <http://www.tei-c.org/Roma>. 2010-10-25.
- TEI Consortium. TEI P5: Guidelines for Electronic Text Encoding and Interchange. Technical report, TEI Consortium, 2010. URL <http://www.tei-c.org/Guidelines/P5/>. 2010-10-25.
- Henry S. Thompson, Murray Maloney, David Beech, and Noah Mendelsohn. XML Schema Part 1: Structures Second series. Recommendation, W3C, 2004. URL <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>. 2010-10-25.
- Norman Walsh. *DocBook 5: The Definitive Guide*. O'Reilly Media and XML Press, 2010.
- Tantek Çelik, Bert Bos, Ian Hickson, and Håkon Wium Lie. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. Candidate Recommendation, W3C, 2009. URL <http://www.w3.org/TR/2009/CR-CSS2-20090908>. 2010-10-25.