Synopsis

## The Story

Since 40 years, our societies have gone through a monumental transformation from print to digital.

This book is a witness view of what happened inside technology and the rippled effects on society as a whole. It is aimed at a general audience, curious to know what happened and how it happened. As much as possible, I will attempt to describe the technological breakthroughs in conceptual terms.

We are not done yet. This book also highlight the unfinished work and address issues that are still pending or unresolved.

## How did it happen?

The society didn't one day decide to abandon paper and turn to digital. It started by a simple modernization process within the publishing industry and ended up impacting most of human activities and the economy.

## Either on or off

Electric devices can be switched on or off. There is usually no intermediary state. Electronic devices are made of myriads of small devices that can be switched on or off.

Letters and numbers are stored as a specific number in a table called a character set. Each character is made of a fixed number of binary numbers, usually 7 or 8. The letter A for example is represented by the number 01000001. Electronically, this means that for this particular character, the first binary digit (bit) is off, the second is on, the others before last are off, and the last one is on, then the character set is a capital A. Pressing the A key on a computer keyboard will create this value, that will be stored in the computer memory, and if a display is connected, the letter "A" will be displayed. If a printer is connected, the letter "A" will be printed.

The difference between any previous writing system and a computerized system is that computers stores values before displaying them.

The printing press was the first machine ever able to mass-produce goods. It was the precursor of the industrial revolution and was also the trigger for massive societal changes, the Renaissance, Enlightenment and the ability for democracies to flourish. Now the time to print was reduced by the ability for computers to produce the photographs that were used instead of the lead plates assembled by hand by typographers who had to pick every character one by one and create the lines. Phototypesetting was implemented massively

during the 1960s and 1970s within dedicated hardware and software, using proprietary coding schemes.

## Computerized text processing

Text writing was formalized by typographical rules that were implemented for printing. Among these rules are the facts that words are separated by spaces, each sentence starts with a capitalized letter and ends with a dot, each paragraph is a set of sentences that starts either with an indentation, or is separated from the previous one by an extra line. Paragraphs can be grouped into sections. Chapters or articles are sets of paragraphs or sections. Supplementary rules exist for footnotes or endnotes, tables of contents, indexes, cross-references.

Pages are created on the fly, while composing a book or an article, and occupy a maximum height, and width. Pages are numbered sequentially, and the page numbers indicate locations in each printed source that are used in tables of contents, indexes, or external references. Pages are "physical" artefacts rather than a logical content structure such as a section.

## Markup: Separation between Structure and Content

On a typewriter, a carriage return is triggered by using a handle that has the effect of advancing the paper to the next line. On an electronic typewriter, a dedicated key triggers a similar action. In a computerized typesetting and word processing machine, a carriage return only amounts to storing a specific code.

Word processors opened the ability for individual users to access features reserved beforehand to professional machines. The word processing systems, like the typesetting systems, rely on a set of codes to trigger actions, such as switching to an alternate font style - bold, italic, etc.- or enlarging or decreasing the font size. Systems were competing according to the number of features they offered, and their internal coding systems varied.

The markup is the set of codes that are provided, in addition to the characters used in the content, to describe formatting changes. In a computerized text stored, the markup is intermixed with the actual content.

The most frequently used brands became reference formats for interchanging documents. Therefore, conversion programs were provided that transformed digitalized text from one format to another. But these programs had to be frequently updated along any changes in the feature list, and they were not guaranteeing a success rate of 100%.

That situation was not sustainable on the long term. Something had to be done. There were two logical ways to go that seemed natural. One way was acknowledging that the industry had to regroup around the most frequently used format and elevate that format at the *de facto* standard. The other way was creating a standard encompassing all possibilities and force all products to adopt that standard.

It turned out that, although these two solutions were actually implemented, an unexpected and innovative alternative was created that turned out to produce a much more significant outcome and eventually revolutionized the world. Before digging into it, let's see how the two "obvious" solutions developed. A group of international experts met at numerous occasions to list all the features existing and that would ever exist in word processing and typesetting and created a standard named "Open Document Architecture". This standard never actually took off.

The word processing industry did consolidate, and Microsoft Word eventually took over its main competitor, WordPerfect, and its .doc format became the preferred way to exchange documents.

That could very well be the end of the story. There were big precedents in the industry of consolidation around the main player, for example the width of the railroad tracks, the electric sockets, screws, etc. However, another alternative approach saw the light of day, and turned out to have a much more significant impact. It is based on the idea that markup is not made of a list of procedural codes, but it can also be used for descriptive purposes. Markup itself could be conceived as a language where users could declare their own requirements. There was no limit to what could be expressed by markup, now left to the imagination and creativity of the users. Instead of using a procedural code such as "italic" to tag the occurrence of a city inside a text, it was possible to declare that the markup tag was going to be "city", and design an independent rendering engine that would render each city name in italics. The decoupling of structure and presentation allowed various rendering engines to operate on the same document. Therefore, with one source, it was possible to produce multiple outputs, for example an electronic version for browsers, and a version aimed at producing a printed document. It became also possible to easily extract data from a document, for example the list of cities, therefore providing at the same time the features needed from documents and from databases.

The Standard Generalized Markup Language (SGML) was published in 1986. It was a framework to define markup languages by declaring not only a set of tags applicable to a document set, but also the rules that the elements marked with the tags should respect. It introduced the use of angle brackets such as London to mark elements within documents. Although angle brackets was a default notation that could be replaced with any other, it was the only one ever used. The work needed to define consistent tag sets, also called document type definitions, was significant, and the work was performed by industry consortiums and were adopted by many companies. The US Government Department of Defense required that its vendors delivered their documents in such a format, and played a big role in the adoption of that standard across many industries. Similar requirements appeared in multiple other countries, mainly in Europe but also in Asia. The Internal Revenue Service was an early adopter as well, and created its own document types for their publications and instructions. An immediate benefit of this approach was that not only the adopters could use the same sources to drive multiple publications, but also that they were guaranteeing that their data was not hostage to a proprietary format, and therefore would survive even if software companies come and go.

The validation process was both a blessing and a curse. SGML-based parsers were able to detect any errors in the document source, and required that all errors be eliminated before going any further. That was exactly what was intended, but the intolerance to even small errors in the markup also slowed down the process and was sometimes considered an unnecessary burden.

At the same time, in the late 1980s, the usage of the Internet was growing. Documents were made available on servers accessible through the network. A new idea, developed by physicists in Geneva, at CERN, led by Tim Berners-Lee, was to link those documents together. He developed a very minimal markup language, inspired by SGML, which featured a minimal number of procedural elements, and, most remarkable to addition of hyperlinks, that allowed documents to call each other, not only at the document level, but even using inner elements as targets. Documents could therefore be interconnected into a web. He created the first software able to display those documents and activate the links between them called a web browser. HTML was not strictly following SGML, as it didn't require documents to be well-formed. The browsers were much more tolerant with markup errors than other SGML-based software. HTML was published in 1991, and was the spark that caused the transition from print to digital to happen and changed the world.

With HTML, it became possible to access the global Internet network, just by clicking on the blue underlined links. The limited set of markup tags made it possible to include headers, lists, tables, and a limited set of typographic variants, such as bold, italic, underline, superscript and subscript. It was also possible to include images within the documents. During a few years, the documents were looking pretty rough, and didn't reach the quality of the printed documents or electronic documents that looked like print (PDF). Over the years, the introduction of a powerful style language (CSS) and the maturation of browsers progressively enhanced the quality of online documents at a point where printed versions became less and less useful.

In the mid 1990s, it also became apparent that SGML was promising too much. It was customizable at a point that nobody needed. For example, nobody cared about using square brackets instead of angle brackets. Also SGML was designed to help users create documents by hand, letting them omit closing tags. As users were using products that were designed to look somewhat like word processors, but creating tags in the background, the need for such shortcuts was considered unnecessary, especially because it was a big hurdle for product vendors that eventually decided that they would only implementing partial features. The result was that documents created with one product were not accepted by another product that implemented a different set of features. Eventually, the extra, not frequently used, features were removed, and a simplified subset of SGML, called XML, was created, and published in 1998. XML became the *lingua franca* for web applications, and was used, not just for documents, but also for software configuration files, database export, and many other applications.

With the advent of the cloud, Web browsers became the ubiquitous way to access information. Like SGML, XML imposed documents to be validated before being used, and that created an obstacle for browsers. Its usage declined strongly, to be replaced by a format, called JSON, that browsers could natively understand, based on Javascript. Despite

the difference in syntax, the information content is very similar, and the principle of using markup to describe information still stands.

Microsoft, which took over the office applications market and became the major force, decided to open their proprietary format for Office documents and store them in XML format (hence the “x” added in the “docx”, “xlsx” or “pptx” file extensions). Even if their internal structure is now public in Office Open XML, its intricate structure still allows Microsoft to maintain its idiosyncratic format that dominates the industry. However, it has facilitated the ability for other similar software to convert from and to their format. A competing product, the open source OpenOffice software suite, uses another XML-based format called “Open Document Format” (ODF).

In 2004, John Gruber created another kind of markup language, called “Markdown”. Markdown is a minimalist language, that contains a limited number of symbols, can be learned in a few minutes, and is now vastly used for blogging, instant messaging, and software documentation. The beauty of Markdown is that texts are still readable with any text editor. The specialized Markdown editors render the documents in a more readable form, but the plain Markdown files are quite readable. Markdown uses # to mark a level 1 header, ## for level 2, etc. A word in italic is surrounded by a star, like here *expression in italic*. Internet links can also be easily created in Markdown, and images can be included. Some more complex features are available in several variants of Markdown, for example tables and footnotes. Markdown is not completely standardized, and there are some variations in the encoding of specialized features that work with specific software packages.