The Digital Printing Press

The history of the Web Site and the Open Source Internet.

Pre WWW (1960's - early 90's)

Early Internet Networks

The Internet has been around long before the web. There are many things that run over the internet, the World Wide Web being one of them. The pre-web internet was largely used by the government and educational institutions to allow complex computer systems to talk to each other.

http://www.ibiblio.org/pioneers/index.html
http://www.w3.org/History.html
http://www.walthowe.com/navnet/history.html

Commercial Networks / Bulletin Board Systems

In the era just before WWW, there were independent commercial networks (like AOL, Prodigy, etc.) that delivered their own content and services to the public, but did not offer access to the internet like the web does. These networks ultimately morphed into WWW ISPs (Internet Service Providors), and opened up to connect to external resources including email and the web.

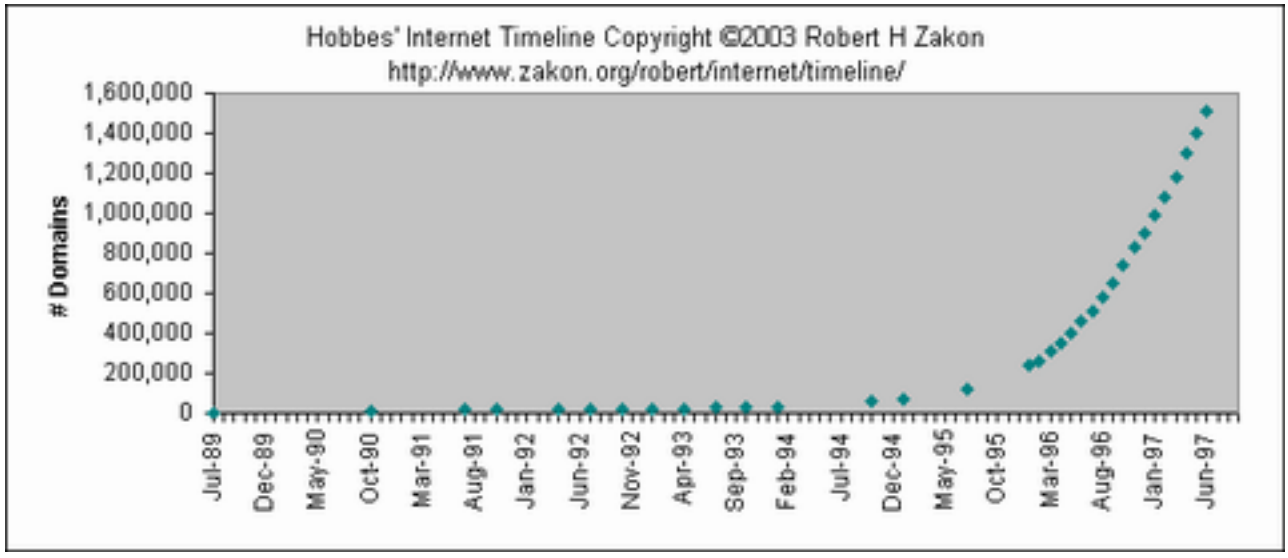
http://www.internetserviceinternetproviders.com/history.asp

The Information Superhighway (Early-Mid 90's)

In 1990 one dude named Tim created the World Wide Web, the first Web Browser, URLs, HTTP and HTML. Then he put it in the public domain and let everyone use it for free. With the help of several different government and non-government organizations, the web was developed in the tech community and rapidly gained support.

HTTP and URLs

The simple way to access website pages (HTTP) made it very easy to create a directory of information that could be accessed through standardized and easy to understand addressing system. Unlike many other internet applications, the addresses for locations on the web (URLs or Domain Names) were made to be easily read by people as well as machines. So Domain Names became not only the access point for a web server (like an IP address) but also an identity, brand name, marketing plan, etc., in a new marketplace that had practically no barriers to entry like other communications markets.



From: http://www.zakon.org/robert/internet/timeline/

The Web Browser:

In 1993 the US National Center for Supercomputing Applications released their Mosaic web browser, which was the most advanced and first widely used application to access the World Wide Web. While the early Internet was used almost exclusively by nerds, the Mosaic browser was the first tool that made the internet accessible by the general public.

Two primary browsers emerged from the Mosaic project, Netscape's Navigator and Microsoft's Internet Explorer. Netscape's Navigator was built by some of the dudes that originally built Mosaic for the NCSA, and had over 90% of the browser market share in its prime.

Only after seeing Navigator's success did Microsoft begin to explore the browser market. Microsoft licensed a version of the Mosaic source code to build Internet Explorer and immediately exploded the browser market. Unlike Navigator, which was a product for sale, Internet Explorer was a free application (for Windows users). With the combination of cheaper internet access fees and Internet Explorer being bundled free with Windows, web surfing soon became a standard aspect of most computing experiences.
http://en.wikipedia.org/wiki/Netscape
http://en.wikipedia.org/wiki/Spyglass,_Inc.d and

HTML

The language used to organize content on web site pages is HTML (Hyper Text Markup Language). HTML describes relationships between pieces of information by grouping them into containers, the same way a printing press organizes content on a newspaper page.

Instead of using two dimensional visual design tools to create a document like in a printing press (or software like photoshop), HTML is a language that describes (or marks up) these relationships in a document structure, or Document Model. This description is interpreted by the browser and the browser relays the translated information to the user on the screen.

These containers (aka tags) wrap around the content and other containers, and represent a tree structure outlining parent and child relationships between these document elements.

```
<div> Hi there </div>


<div>
        Hi there <span font-color="green"> Steve </span>
        <img src="image_link_url/image.jpg" alt="alternate_image_title"
/>
        <a href="hyperlink_url" target="_blank">Link Text</a>
       <form>
          <input type="text" />
          <input type="checkbox">
          <input type="submit" />
    </form>

</div>
```

In it's early form, HTML served a simple set of functions:

- displaying text, data sets, and images
- "hyperlinking" to other pages
- accepting and sending data from forms
- presenting page layout, colors, and typography
- describing properties of the page to the internet and to the browser

The Document Model

Any object that can be built has a model, from a web page to a chair to a spaceship. The model represents all of the rules that need to be met for that object to serve its purpose. The blueprints for an object need to be designed around this model in order for the object to work.

For a building object, blueprint designs are written to set up all of the properties of the building. Then an engineer takes the blueprint, acquires all of the materials needed from the blueprint, and constructs the building. The building object's model includes zoning regulations, the laws of gravity, and the specific needs of the building owner. Those specific needs also have models that they have to follow.

In a web page document, the Web Browser takes blueprints (HTML) and builds the document object's structure that contains the content. The document model includes the rules for interpreting different document properties and how the HTML must be structured to render correctly.

It also has rules for a number of other purposes, including how the document objects can be accessed with browser scripts.

-- DOM TREE IMAGE --

Browser Scripting, pre-DOM

Although HTML provided some interactivity with links and forms, web developers needed to provide other interactive functionality on the page. So different (browser specific) scripting languages evolved in order to access and manipulate objects in the document, or (DOM objects).

These languages use events in the browser to trigger functions that, in turn, execute other events. So just as the server can run functions to generate and change what you see on the page, this capability could also be done after the information was sent to the browser.

Early HTML & Browser Challenges:

Early web browsers developed different ways to interpret the objects in a document with non-standardized document models.

The use of HTML tags was not very consistent from one website to the next or from one browser to the next. Specific browsers began to implement their own tags to add functionality to early HTML. Also, it was possible generate the same thing on a web page in numerous different ways that were not necessarily the intent of the HTML language. So using HTML markup to describe the content's meaning largely took a backseat to using HTML tags for design purposes. For example, HTML tables (which were intended to be used like in Excel - to represent sets of data) were used across the board to create page structure (header, content area, footer, etc).

Tim built the WorldWideWeb appliction in mind so that scientists could share documents with other scientists even if they were on different software platforms. Not a whole lot of concern was put into this technology being used for designing layouts.

Also font size, colors, widths, height, were described as attributes of a specific HTML element. This cluttered up the code and made it inefficient to manage the design across multiple pages. Changing something on 20 different pages meant changing it 20 times. http://www.yourhtmlsource.com/starthere/historyofhtml.html

The scripting languages were written from of the same original code (EMCAscript), but the two main browsers developed their own variations from this standard. While it allowed developers to build interactivity into their documents, the inconsistencies between these the two main platforms (IE & Navigator) required writing convoluted code in which the same functionality often had to be written multiple times. This period was referred to as the (First) "Browser Wars" because the two primary browser engines were attempting to pull the web surfing experience into different directions.

It was soon clear there needed to be a set of standards for how to interpret and implement elements in the Document Model. The first purpose of the browser was simply to allow humans to interpret data, but the web would soon depend on other types of non human interpreters to also decipher this information (search engines, rich internet applications). These interpretations would require a much more strict set of rules for those machines to understand.

The W3C, Web Standards, and the DOM (Mid-Late 90's)

The solution to these problems began with a collective effort to implement a standardized convention for how HTML elements and page functions were interpreted by browsers. A group called the World Wide Web Consortium (aka W3C, also started by Tim) organized with the purpose of developing a recommended set of standards for using these technologies. Along with developing standards for HTML and CSS syntax, the W3C most importantly developed a standardized Document Model to represent the rules for how browsers should interpret all document functions (including HTML and Browser Scripting).

This set of rules came to be known as the Document Object Model, or DOM. The standardized DOM turned the browser from primarily a content reader into a standardized platform that could be used to create complex interactive applications.

When each browser had its own rules, it was pointless to rely on the browser technology to process complex logic when the server technology was consistent and thereby far more reliable. Plus any software made to run on the browser was at the mercy of the browser companies' proprietary technology. If Microsoft was able to control the browser market and force developers to use its JScript, then companies like Google would be completely dependent on Microsoft's decisions on how they can use their code.

However with a standard set of rules, server software could rely on any standards compliant browser to process large parts of its functionality. The myriad of Google apps like Gmail, Google Maps, Google Docs, etc. all rely on the these advanced processing capabilities to do a lot of the fancy stuff we see.

Because it was more in line with this new DOM specification, and was open source, Javascript ultimately beat out Microsoft's Jscript and is now the standard browser scripting language.

Netscape goes Open Source (1998)

While the web and web browsers relied on numerous open source / public domain technologies (from Linux to HTML to Javascript), the use of proprietary systems to control the browser market threatened the growth of new web technologies. Netscape and Microsoft were going in different directions, which was resulting in a less than optimal web experience.

Beign a much smaller company than Microsoft, Netscape was unable to compete with Microsoft's commanding market share and found itself unable to make a profit. Before getting bought out by AOL, Netscape decided to release their source code to the public, allowing anyone to build browser technologies from their specs.
http://www-archive.mozilla.org/src-faq.html

Along with the rise of high-speed internet, Netscape's move to open source proved to be one of the most significant breakthroughs in the development of next generation of web technologies. Opening Netscape's code led the way to a diverse range of new browser platforms that were both open source and supported the web standards recommended by the W3C.

This was much to the disadvantage of Microsoft, who's overall strategy with Internet Explorer was to create a dominant proprietary browser experience that would hinder the possibility of developing viable alternatives. But Netscape's move would ultimately cause Microsoft to loose their dominant position due to much more competition in the browser market.

CSS (late 90s - early 2000's)

In the same fashion that the DOM was developed, a set of standardized style attributes were conceived that came to be known as Cascading Style Sheets. These removed the need for these attributes in HTML tags. This now allowed website builders to reference style properties remotely. Attributes could now be attached to the HTML DOM elements from an outside source, allowing them to be styled globally or in specific groups.

This also eliminated the need to use HTML elements for layout purposes. CSS was able to represent those relationships while **keeping the HTML elements free to do what they were intended to do, which is to describe content purpose.**

Similar to it's scripting language, Microsoft chose to set up its own variation of these style rules to in part make it difficult for competing open source alternatives to operate. Microsoft's interpretation of CSS seemed to provide no real advantage except to serve its competitive interests, and has ultimately only led to a contempt for the IE browser by anyone that writes CSS. However, Microsoft has lately (albeit very reluctantly) moved towards implementing many of the W3C's recommendations, including those for CSS.

Firefox, Plugins, and Firebug! (2000's)

When Netscape went open source, a company called Mozilla was started to develop open access software in part from Netscape Navigator's operating platform. Being open source, Mozilla's greatest benefit over Microsoft's model was that the web development community had a transparent view of Mozilla's projects and access to their source code.

Microsoft's closed-source model requires that all development, testing, and fixes to be done by their internal technology team, whereas Mozilla's products are supported by the community that uses the application. If a security threat is identified in IE, it requires Microsoft to asses the situation and come up with the solution and then submit the fix to users. If a security threat is identified in a Mozilla project, anyone that wants to mess with the code can attempt to identify the bug, find a solution, and publish documentation on how to patch the error until it is fixed in the source code.

Mozilla's web browser - Firefox - became a strong competitor to IE by pushing forward the capabilities of what a browser was expected to do. Aside from tabbed browsing, the most significant difference between the new Firefox and IE was that Firefox allowed the development community to build mini-applications within the browser platform to add

special functionality. These Plugins allow users to access and manipulate data from the web in any number of ways beyond what the browser did by itself.

One of the most significant plugins for developing websites is the Firebug browser testing tool. Firebug helps identify all the things going on behind the scenes that make the DOM spit out what you see on the page. This and similar javascript-made tools have led to much improved front end coding practices. With Firebug, a developer can immediately see the result of a DOM event or page change, instead of making an educated guess (which was generally the way front end debugging was done prior to having these tools). getfirebug.com

XML & Ajax (early/mid 2000's)

XML is a language that organizes sets of data in a similar way way you would in a database or excel spreadsheet. XML uses the same Document Object Model to process data that HTML uses to render page elements. The same way HTML tags reference different types of elements on a page (links, images, paragraphs, page header), XML tags reference different elements of data in a of data set (album title, artist, songs, image, release date, etc.).

```
<album>
        <artist>artistname</artist>
        <title>albumtitle</title>
        <playlist>
                <track1>name</track1>
                <track2>name</track2>
        </playlist>
</album>
```

The tag names are created by the user and identify relationships between the different pieces of content, but say nothing of how they get displayed.

XML is often used to send data "feeds" (RSS) from one application to another or straight to the browser. The data is sent without any presentation properties (HTML), allowing whoever is using the data to capture it and then present it in whatever HTML format they choose.

One of the benefits of XML is that other DOM based languages (like Javascript) can be used on XML in similar ways. This is the foundation of the concept of AJAX (or Asynchronous Javascript And XML). AJAX is not a language, but a type of javascript function that sends and receives data back and forth to the server as part of the page functionality. Without AJAX, the browser would need to re-access the whole page from the server each time it wanted to get or send new content to/from the server. But with AJAX, data can be sent back and forth between the browser and the server without reloading the whole page.

Whether it is from a remote source or within an application, an XML document can be used with AJAX to update content on a page after the main page load. Javascript can parse the XML data, grab the relevant elements, build HTML just for those elements, and plug it into the page dynamically.

While XML is a solid language and well supported, JSON (Javascript Object Notation) seems to be a more lean and extensible option for data organization and may overtake XML as a preferred client side data format. XML parsing languages such as XSLT are also relatively outdated as much of this can now be done with the same Javascript and Ajax libraries that provide a much wider range of functionality.

XHTML (mid 2000's)

Both XML and HTML are Markup Languages (the ML part) meaning their main purpose is to mark-up or format data. XHTML's goal was to combine the functionality of XML with XHTML so they could serve the purposes of both organizing data and presentation.

One attempt of implementing XHTML standards was to allow HTML documents to be interpreted as data sources, in the same way that XML is used. I am not sure what the goal was here, however this is not what HTML pages were meant to do and this functionality was never really used for anything. So in that aspect, XHTML validation serves little more purpose than regular HTML validation. The only exception being that I would say it does help front-end web developers follow a better structured coding convention.

While XHTML will eventually give way to the rules for the HTML5, the main contribution of XHTML to markup language is the use of semantic naming conventions to describe pieces of code. XHTML promoted a way to represent this relationship with HTML classes the same way that tags are used in XML. For example, the album XML markup above can be represented the same way in an XHTML document like this.

```
<div class="album">
        <span class="artist">Artist Name</span>
        <span class="title">Album Title</span>
        <ol class="playlist">
                <li>Track 1</li>
                <li>Track 2</li>
        </ol>
</div>
```

The <ol> tag is an ordered list with <li> list items that will be displayed as 1. track1, 2. track2, etc.

Now this piece of code not only dictates the presentation of the content but also represents its meaning.

Semantic coding like this has led to the emergence of much more clean and well structured CSS & Javascript practices (especially with the use of javascript libraries such as jQuery).

Javascript Libraries (late 2000's)

Numerous Javascript libraries emerged in the mid/late 2000's which created simple exstensible frameworks to execute specific types of Javascript events. No different than texting acronyms or transcribing shorthand, javascript libraries create simplified abbreviated syntax based off the original language. jQuery is arguably the most popular of the open source Javascript libraries at the moment, however Google, Yahoo, and several other groups also have developed powerful open source libraries as well.

Modern Web Enabled Devices (Late 2000's)

With the advent of smartphones and 3G networks, the cell phone has become as much of an access point to the internet as our computers. Much like early web browsers, there has been a significant level of inconsistency across different mobile devices. However the next generation smart phones (like the iPhone and the Google Android) have already incorporated the same modern browser technologies used in desktop machines. Next generation mobile devices such as the iPhone, iPad and Kindle already are moving the computing experience to a more natural, less technical interaction with web applications.

Looking several years down the road, mobile devices will certainly shape the user experience more than any other computing use. Advanced functionality beginning to be

used in mobile devices like touch screens, voice recognition, etc., will completely shift the experience into an even more natural "human experience" where the user will interact with the browser more the way that one would interact with a person or non-computer device.

HTML5 / CSS3 (2010's)

The next generation standards being developed by the W3C for HTML and CSS will create an even more enhanced web experience with an extensive array of advanced functions and more intuitive programming uses.
http://dev.w3.org/html5/html4-differences/

Looking Back

All of these advancements in web technology can largely be traced back to the successful adoption of a standardized Document Model for generating web pages. While software platforms like .NET, Java, Ruby, PHP etc., continue to compete for server side dominance, the technologies that create the rich user experience of the modern World Wide Web have evolved due in large part to these open source DOM standards.

It can be argued that that the success of the World Wide Web is generally synonymous with the success of open source technology. These technologies have made the internet experience both affordable and accessible for the general public.

- GNU/LINUX servers run a lot of the internet
- The entire realm of WorldWideWeb technologies maintained by the W3C is not just open source but exists completely in the public domain. That is: WWW, HTTP, HTTPS, URLs, the DOM, Javascript, HTML, CSS, XML, and much more.
- Open source browser technology (Mozilla, Webkit, Gecko) has significantly enhanced the web surfing experience. During Microsoft's IE most dominant period in the market, this progress was certainly much slower.

While Microsoft ultimately failed at cornering the browser market, it's strategy to release its own browser as freeware played one of the most significant roles in shaping the web experience we have today. It brought the web to the public a whole lot quicker, but ultimately also destroyed its own competitive business model and moved web browser technology into the public sphere where it has thrived.

Because of Microsoft's position in the market, it became almost impossible for Netscape and others to compete on the same playing field. But with no competitive options left, Netscape's decision to release it's code to the public was the only thing that was able to level the playing field to counter Microsoft's dominance.

At the beginning, Netscape was forced to charge for their product, as there was no other business model to make profit off of the web like there is now. On the other side, Microsoft had no need to make money from its browser, since its goal was clearly more to incorporate the browsing experience into its Windows operating system. Microsoft's competitive nature led it to defy the W3C's recommendations at times in order to disrupt the potential for compatible browsers to provide the same user experience. Though their hope of drowning out competition did not work, and ultimately only served to make things more complicated for the web development community which pushed the need for a more reliable approach.

Unfortunately for Microsoft, their early web strategy may ultimately prove to be their biggest failure, and clearly why they have lost their dominant place in the tech world. Though Microsoft was ultimately able to put it's main competitor out of business, Netscape's influence will very likely outlast Microsoft's as open source technology will continue to be the driving force behind the modern web experience.