Web Engineering

Abby Tucker

First Edition, 2012

ISBN 978-81-323-3114-8

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Introduction

The World Wide Web have become a major delivery platform for a variety of complex and sophisticated enterprise applications in several domains. In addition to their inherent multifaceted functionality, these web applications exhibit complex behavior and place some unique demands on their usability, performance, security and ability to grow and evolve. However, a vast majority of these applications continue to be developed in an ad-hoc way, contributing to problems of usability, maintainability, quality and reliability. While web development can benefit from established practices from other related disciplines, it has certain distinguishing characteristics that demand special considerations. In the recent years, there have been some developments towards addressing these problems and requirements. As an emerging discipline, **web engineering** actively promotes systematic, disciplined and quantifiable approaches towards successful development of high-quality, ubiquitously usable web-based systems and applications. In particular, web engineering focuses on the methodologies, techniques and tools that are the foundation of web application development and which support their design, development, evolution, and evaluation. Web application development has certain characteristics that make it different from traditional software, information system, or computer application development.

Web engineering is multidisciplinary and encompasses contributions from diverse areas: systems analysis and design, software engineering, hypermedia/hypertext engineering, requirements engineering, human-computer interaction, user interface, information engineering, information indexing and retrieval, testing, modelling and simulation, project management, and graphic design and presentation. Web engineering is neither a clone, nor a subset of software engineering, although both involve programming and software development. While web Engineering uses software engineering principles, it encompasses new approaches, methodologies, tools, techniques, and guidelines to meet the unique requirements of web-based applications.

Web engineering as a discipline

Proponents of web engineering supported the establishment of web engineering as a discipline at an early stage of web. First Workshop on Web Engineering was held in conjunction with World Wide Web Conference held in Brisbane, Australia, in 1998. San

Murugesan, Yogesh Deshpande, Steve Hansen and Athula Ginige, from University of Western Sydney, Australia formally promoted web engineering a new discipline in the first ICSE workshop on Web Engineering in 1999. Since then they published a serial of papers in a number of journals, conferences and magazines to promote their view and got wide support. Major arguments for web engineering as a new discipline are:

- and Web-based Information Systems (WIS) development process are different and unique.
- Web engineering is multi-disciplinary; no single discipline (such as software engineering) can provide complete theory basis, body of knowledge and practices to guide WIS development.
- Issues of evolution and lifecycle management when compared to more 'traditional' applications.
- Web based information systems and applications are pervasive and non-trivial. The prospect of web as a platform will continue to grow and it is worth being treated specifically.

However, it has been controversial, especially for people in other traditional disciplines such as software engineering, to recognize web engineering as a new field. The issue is how different and independent web engineering is, compared with other disciplines.

Main topics of Web engineering include, but are not limited to, the following areas:

Web Process & Project Management Disciplines

- Development Process and Process Improvement of Web Applications
- Web Project Management and Risk Management
- Collaborative Web Development

Web Requirements Modeling Disciplines

- Business Processes for Applications on the Web
- Process Modelling of Web applications
- Requirements Engineering for Web applications

Web System Design Disciplines, Tools & Methods

- UML and the Web
- Conceptual Modeling of Web Applications (aka. Web modeling)
- Prototyping Methods and Tools
- Web design methods
- CASE Tools for Web Applications
- Web Interface Design
- Data Models for Web Information Systems

Web System Implementation Disciplines

- Integrated Web Application Development Environments

- Code Generation for Web Applications
- Software Factories for/on the Web
- Web 2.0, AJAX, E4X, Asp.net2.0,Asp.net3.0 and Other New Developments
- Web Services Development and Deployment
- Empirical Web Engineering

Web System Testing Disciplines

- Testing and Evaluation of Web systems and Applications
- Testing Automation, Methods and Tools

Web Applications Categories Disciplines

- Semantic Web applications
- Ubiquitous and Mobile Web Applications
- Mobile Web Application Development
- Device Independent Web Delivery
- Localization and Internationalization Of Web Applications

Web Quality Attributes Disciplines

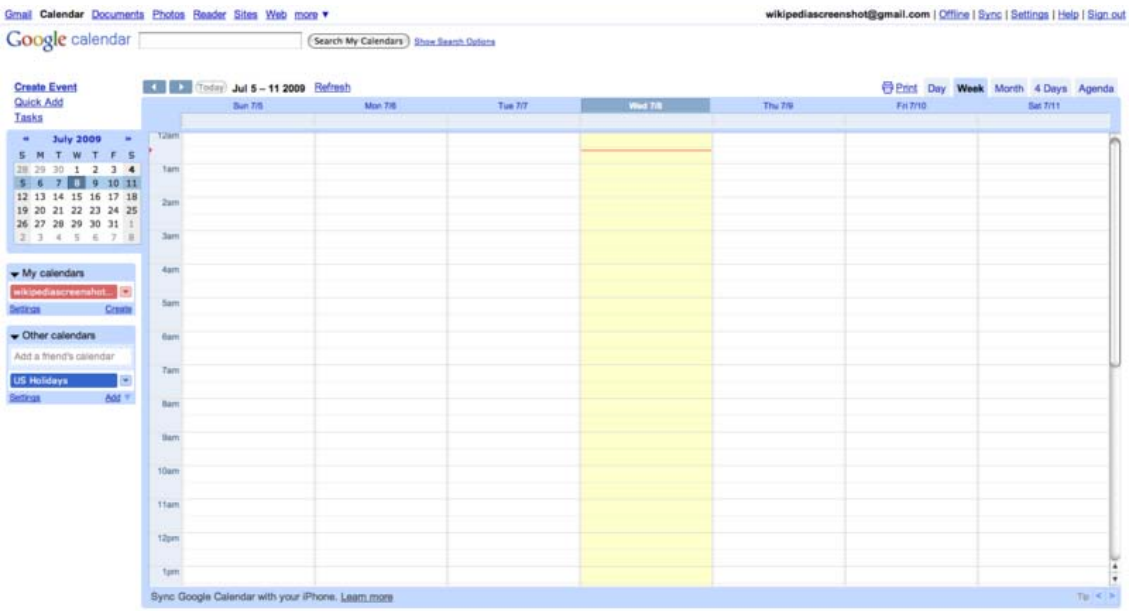
- Web Metrics, Cost Estimation, and Measurement
- Personalisation and Adaptation of Web applications
- Web Quality
- Usability of Web Applications
- Web accessibility
- Performance of Web-based applications

Content-related Disciplines

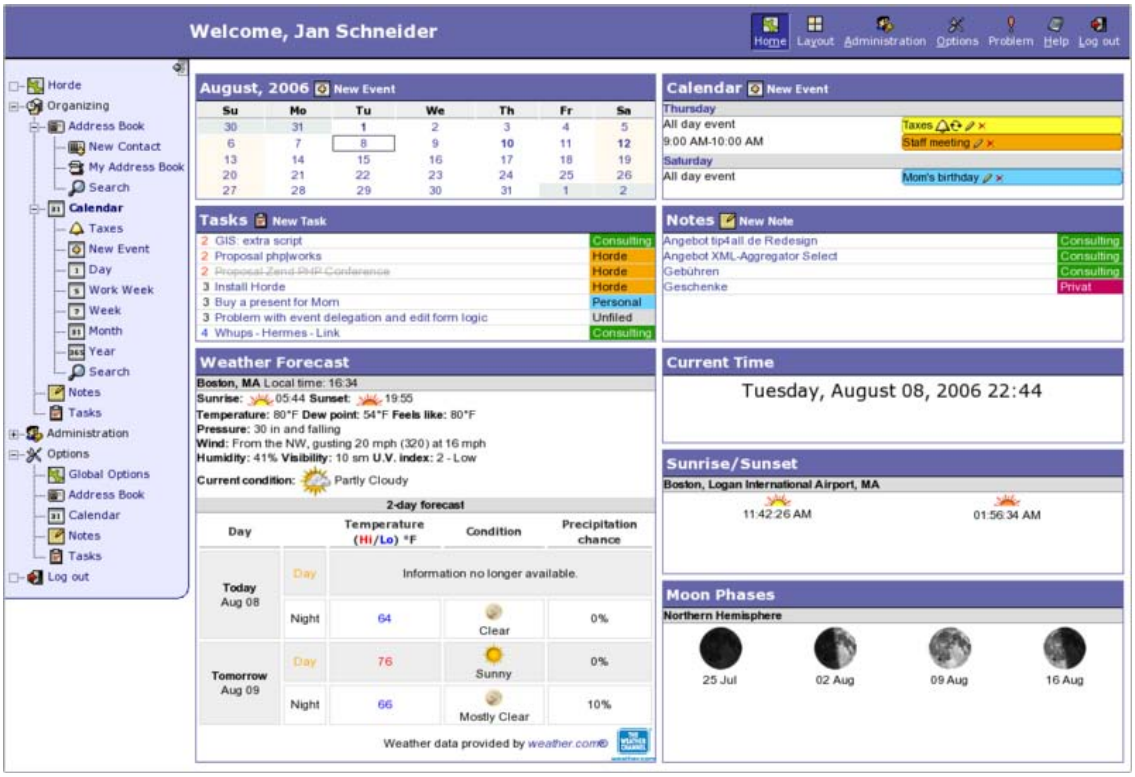
- Web Content Management
- Multimedia Authoring Tools and Software
- Authoring of adaptive hypermedia

Chapter 1

Web Application



Google Calendar is a contact- and time-management web application offered by Google.



Horde groupware is an open source web application.

A **web application** is an application that is accessed over a network such as the Internet or an intranet. The term may also mean a computer software application that is hosted in a browser-controlled environment (e.g. a Java applet) or coded in a browser-supported language (such as JavaScript, combined with a browser-rendered markup language like HTML) and reliant on a common web browser to render the application executable.

Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions and many other functions.

History

In earlier computing models, e.g. in client-server, the load for the application was shared between code on the server and code installed on each client locally. In other words, an application had its own client program which served as its user interface and had to be separately installed on each user's personal computer. An upgrade to the server-side code of the application would typically also require an upgrade to the client-side code installed on each user workstation, adding to the support cost and decreasing productivity.

In contrast, web applications use web documents written in a standard format such as HTML (and more recently XHTML), which are supported by a variety of web browsers.

Generally, each individual web page is delivered to the client as a static document, but the sequence of pages can provide an interactive experience, as user input is returned through web form elements embedded in the page markup. During the session, the web browser interprets and displays the pages, and acts as the *universal* client for any web application.

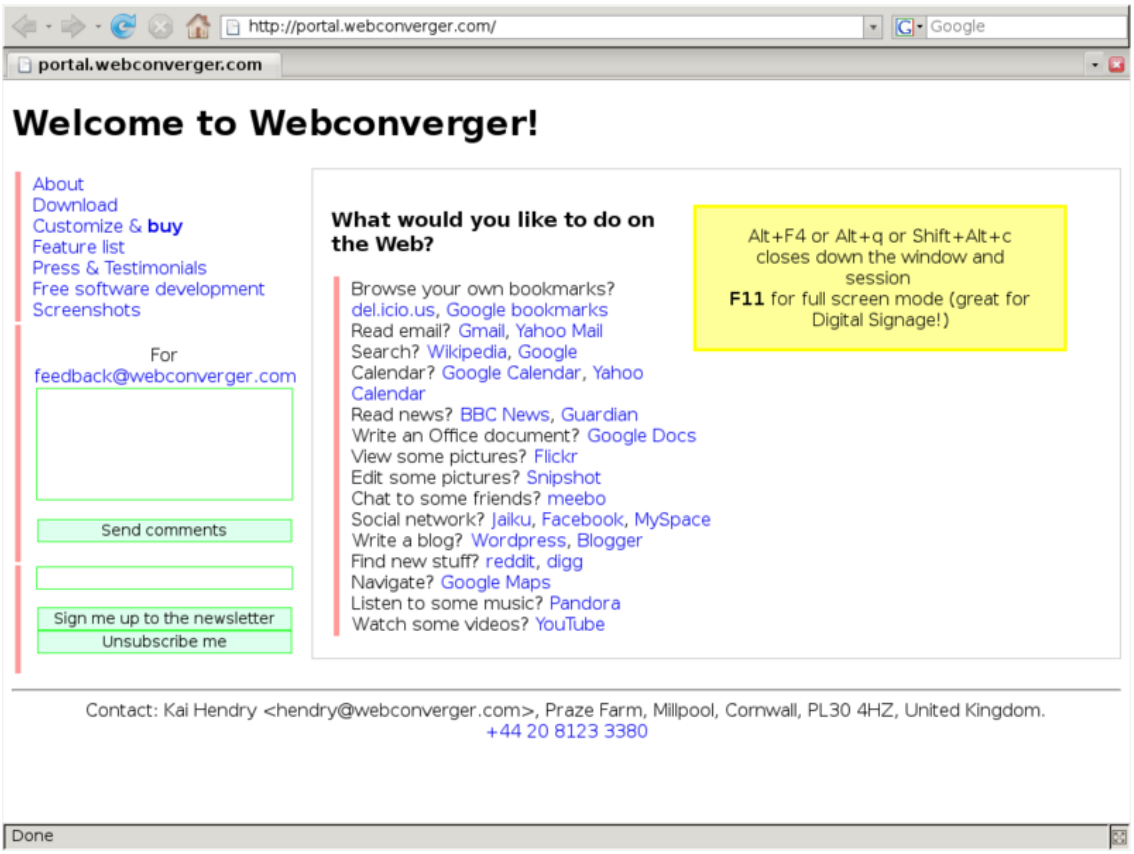
In 1995, Netscape introduced a client-side scripting language called JavaScript, which allowed programmers to add some dynamic elements to the user interface that ran on the client side. Until then, all the data had to be sent to the server for processing, and the results were delivered through static HTML pages sent back to the client

In 1996, Macromedia introduced Flash, a vector animation player that could be added to browsers as a plug-in to embed animations on the web pages. It allowed the use of a scripting language to program interactions on the client side with no need to communicate with the server.

In 1999, the "web application" concept was introduced in the Java language in the Servlet Specification version 2.2. [2.1?]. At that time both JavaScript and XML had already been developed, but Ajax had still not yet been coined and the XMLHttpRequest object had only been recently introduced on Internet Explorer 5 as an ActiveX object.

In 2005, the term Ajax was coined, and applications like Gmail started to make their client sides more and more interactive.

Interface



Webconverger operating system provides an interface for web applications.

Through Java, JavaScript, DHTML, Flash, Silverlight and other technologies, application-specific methods such as drawing on the screen, playing audio, and access to the keyboard and mouse are all possible. Many services have worked to combine all of these into a more familiar interface that adopts the appearance of an operating system. General purpose techniques such as drag and drop are also supported by these technologies. Web developers often use client-side scripting to add functionality, especially to create an interactive experience that does not require page reloading. Recently, technologies have been developed to coordinate client-side scripting with server-side technologies such as PHP. Ajax, a web development technique using a combination of various technologies, is an example of technology which creates a more interactive experience.

Structure

Applications are usually broken into logical chunks called "tiers", where every tier is assigned a role. Traditional applications consist only of 1 tier, which resides on the client machine, but web applications lend themselves to a n-tiered approach by nature. Though many variations are possible, the most common structure is the three-tiered application. In its most common form, the three tiers are called *presentation*, *application* and *storage*,

in this order. A web browser is the first tier (presentation), an engine using some dynamic Web content technology (such as ASP, ASP.NET, CGI, ColdFusion, JSP/Java, PHP, Perl, Python, Ruby on Rails or Struts2) is the middle tier (application logic), and a database is the third tier (storage). The web browser sends requests to the middle tier, which services them by making queries and updates against the database and generates a user interface.

For more complex applications, a 3-tier solution may fall short, and you may need a n-tiered approach, where the greatest benefit is breaking the business logic, which resides on the application tier, into a more fine-grained model. Or adding an integration tier that separates the data tier from the rest of tiers by providing an easy-to-use interface to access the data. For example, you would access the client data by calling a "list_clients()" function instead of making a SQL query directly against the client table on the database. That allows you to replace the underlying database without changing the other tiers.

There are some who view a web application as a two-tier architecture. This can be a "smart" client that performs all the work and queries a "dumb" server, or a "dumb" client that relies on a "smart" server. The client would handle the presentation tier, the server would have the database (storage tier), and the business logic (application tier) would be on one of them or on both. While this increases the scalability of the applications and separates the display and the database, it still doesn't allow for true specialization of layers, so most applications will outgrow this model.

Business use

An emerging strategy for application software companies is to provide web access to software previously distributed as local applications. Depending on the type of application, it may require the development of an entirely different browser-based interface, or merely adapting an existing application to use different presentation technology. These programs allow the user to pay a monthly or yearly fee for use of a software application without having to install it on a local hard drive. A company which follows this strategy is known as an application service provider (ASP), and ASPs are currently receiving much attention in the software industry.

Writing web applications

There are many web application frameworks which facilitate rapid application development by allowing the programmer to define a high-level description of the program. In addition, there is potential for the development of applications on Internet operating systems, although currently there are not many viable platforms that fit this model.

The use of web application frameworks can often reduce the number of errors in a program, both by making the code simpler, and by allowing one team to concentrate just on the framework. In applications which are exposed to constant hacking attempts on the Internet, security-related problems can be caused by errors in the program. Frameworks can also promote the use of best practices such as GET after POST.

Applications

Browser applications typically include simple office software (word processors, online spreadsheets, and presentation tools), with Google Docs being the most notable example, and can also include more advanced applications such as project management, computer-aided design, video editing and point-of-sale.

Benefits

- Web applications do not require any complex "roll out" procedure to deploy in large organizations. A compatible web browser is all that is needed;
- Browser applications typically require little or no disk space on the client;
- They require no upgrade procedure since all new features are implemented on the server and automatically delivered to the users;
- Web applications integrate easily into other server-side web procedures, such as email and searching.
- They also provide cross-platform compatibility in most cases (i.e., Windows, Mac, Linux, etc.) because they operate within a web browser window.

Drawbacks

- In practice, web interfaces, compared to thick clients, typically force significant sacrifice to user experience and basic usability.
- Web applications absolutely require compatible web browsers. If a browser vendor decides not to implement a certain feature, or abandons a particular platform or operating system version, this may affect a huge number of users;
- Standards compliance is an issue with any non-typical office document creator, which causes problems when file sharing and collaboration becomes critical;
- Browser applications rely on application files accessed on remote servers through the Internet. Therefore, when connection is interrupted, the application is no longer usable but if it uses HTML5 API's such as Offline Web application caching, it can be downloaded and installed locally, for offline use. Google Gears, although no longer in active development, is a good example of a third party plugin for web browsers that provides additional functionality for creating web applications;
- Since many web applications are not open source, there is also a loss of flexibility, making users dependent on third-party servers, not allowing customizations on the software and preventing users from running applications offline (in most cases). However, if licensed, proprietary software can be customized and run on the preferred server of the rights owner;
- They depend entirely on the availability of the server delivering the application. If a company goes bankrupt and the server is shut down, the users have little recourse. Traditional installed software keeps functioning even after the demise of the company that produced it (though there will be no updates or customer service);
- Likewise, the company has much greater control over the software and functionality. They can roll out new features whenever they wish, even if the

users would like to wait until the bugs have been worked out before upgrading. The option of simply skipping a weak software version is often not available. The company can foist unwanted features on the users or cut costs by reducing bandwidth. Of course, companies will try to keep the good will of their customers, but the users of web applications have fewer options in such cases unless a competitor steps in and offers a better product and easy migration;

- The company can theoretically track anything the users do. This can cause privacy problems.

Chapter 2

Web Design

An example of a web page that uses CSS Layouts

Web design is a broad term used to encompass the way that content (usually hypertext or hypermedia) is delivered to an end-user through the World Wide Web, using a web browser or other web-enabled software is displayed. The intent of web design is to create a website—a collection of online content including documents and applications that reside on a web server/servers. A website may include text, images, sounds and other content, and may be interactive.

Overview

Web design involves the structure of the website including the information architecture (navigation schemes and naming conventions), the layout or the pages (wireframes or page schematics are created to show consistent placement of items including functional features), and the conceptual design with branding.

Content

Such elements as text, forms, images (GIFs, JPEGs, Portable Network Graphics) and video can be placed on the page using HTML/XHTML/XML tags. Older browsers may require Plug-ins such as Adobe Flash, QuickTime, Java run-time environment, etc. to display some media, which are embedded into web page by using HTML/XHTML tags.

Improvements in browsers' compliance with W3C standards prompted a widespread acceptance and usage of XHTML/XML in conjunction with Cascading Style Sheets (CSS) to position and manipulate web page elements and objects.

Typically web pages are classified as static or dynamic:

- Static pages don’t change content and layout with every request unless a human (web master/programmer) manually updates the page. A simple HTML page is an example of static content.
- Dynamic pages adapt their content and/or appearance depending on end-user’s input/interaction or changes in the computing environment (user, time, database modifications, etc.) Content can be changed on the client side (end-user's computer) by using client-side scripting languages (JavaScript, JScript, Actionscript, etc.) to alter DOM elements (DHTML). Dynamic content is often compiled on the server utilizing server-side scripting languages (Perl, PHP, ASP, JSP, ColdFusion, etc.). Both approaches are usually used in complex applications.

With growing specialization in the information technology field there is a strong tendency to distinguish between web design and web development. Web design is a kind of graphic design intended for the development and styling of objects of the Internet's information environment to provide them with high-end consumer features and aesthetic qualities.

This definition separates web design from web programming, emphasizing the functional features of a web site, as well as positioning web design as a kind of graphic design. The process of designing web pages, web sites, web applications or multimedia for the Web

may utilize multiple disciplines, such as animation, authoring, communication design, corporate identity, graphic design, human-computer interaction, information architecture, interaction design, marketing, photography, search engine optimization and typography.

- Markup languages (such as HTML, XHTML and XML)
- Style sheet languages (such as CSS and XSL)
- Client-side scripting (such as JavaScript)
- Server-side scripting (such as PHP and ASP)
- Database technologies (such as MySQL and PostgreSQL)
- Multimedia technologies (such as Flash and Silverlight)

Web pages and websites can be static pages, or can be programmed to be dynamic pages that automatically adapt content or visual appearance depending on a variety of factors, such as input from the end-user, input from the webmaster or changes in the computing environment (such as the site's associated database having been modified).

Accessible web design

To be accessible, web pages and sites must conform to certain accessibility principles. These accessibility principles are known as the WCAG when talking about content. These can be grouped into the following main areas.

- Use semantic markup that provides a meaningful structure to the document (i.e. web page)
- Semantic markup also refers to semantically organizing the web page structure and publishing web services description accordingly so that they can be recognized by other web services on different web pages. Standards for semantic web are set by IEEE
- Use a valid markup language that conforms to a published DTD or Schema
- Provide text equivalents for any non-text components (e.g. images, multimedia)
- Use hyperlinks that make sense when read out of context. (e.g. avoid "Click Here")

Website accessibility is also changing as it is impacted by Content Management Systems that allow changes to be made to webpages without the need of obtaining web-based programming language knowledge.

It is very important that several different components of web development and interaction can work together in order for the Web to be accessible to people with disabilities. These components include:

- content - the information in a web page or web application, including:
  - natural information such as text, images, and sounds
  - code or markup that defines structure, presentation, etc.
- Web browsers, media players, and other "user agents"
- assistive technology, in some cases - screen readers, alternative keyboards, switches, scanning software, etc.

- users' knowledge, experiences, and in some cases, adaptive strategies using the Web
- developers - designers, coders, authors, etc., including developers with disabilities and users who contribute content
- authoring tools - software that creates web sites
- evaluation tools - web accessibility evaluation tools, HTML validators, CSS validators, etc.

History

Tim Berners-Lee published what is considered to be the first website in August 1991. Berners-Lee was the first to combine Internet communication (which had been carrying email and the Usenet for decades) with hypertext (which had also been around for decades, but limited to browsing information stored on a single computer, such as interactive CD-ROM design). Websites are written in a markup language called HTML, and early versions of HTML were very basic, only giving a website's basic structure (headings and paragraphs), and the ability to link using hypertext. This was new and different from existing forms of communication - users could easily navigate to other pages by following hyperlinks from page to page.

As the Web and web design progressed, the markup language changed to become more complex and flexible, giving the ability to add objects like images and tables to a page. Features like tables, which were originally intended to be used to display tabular information, were soon subverted for use as invisible layout devices. With the advent of Cascading Style Sheets (CSS), table-based layout is commonly regarded as outdated. Database integration technologies such as server-side scripting and design standards like W3C further changed and enhanced the way the Web is made. As times change, websites are changing the code on the inside and visual design on the outside with ever-evolving programs and utilities.

With the progression of the Web, tens of thousands of web design companies have been established around the world to serve the growing demand for such work. As with much of the information technology industry, many web design companies have been established in technology parks in the developing world as well as many Western design companies setting up offices in countries such as India, Romania, and Russia to take advantage of the relatively lower labor rates found in such countries.

Website planning

Purposing web design is a complex, but essential ongoing activity. Before creating and uploading a website, it is important to take the time to plan exactly what is needed in the website. Thoroughly considering the audience or target market, as well as defining the purpose and deciding what content will be developed, are extremely important.

Context

Web design is similar (in a very simplistic way) to traditional print publishing. Every website is an information display container, just as a book; and every web page is like the page in a book. However, web design uses a framework based on digital code and display technology to construct and maintain an environment to distribute information in multiple formats. Taken to its fullest potential, web design is undoubtedly the most sophisticated and increasingly complex method to support communication in today's world.

Purpose

It is essential to define the purpose of the website as one of the first steps in the planning process. A purpose statement should show focus based on what the website will accomplish and what the users will get from it. A clearly defined purpose will help the rest of the planning process as the audience is identified and the content of the site is developed. Setting short and long term goals for the website will help make the purpose clear and plan for the future when expansion, modification, and improvement will take place. Measurable objectives should be identified to track the progress of the site and determine success.

Audience

Defining the audience is a key step in the website planning process. The audience is the group of people who are expected to visit your website – the market being targeted. These people will be viewing the website for a specific reason and it is important to know exactly what they are looking for when they visit the site. A clearly defined purpose or goal of the site as well as an understanding of what visitors want to do or feel when they come to your site will help to identify the target audience. Upon considering who is most likely to need or use the content, a list of characteristics common to the users such as:

- Audience Characteristics
- Information Preferences
- Computer Specifications
- Web Experience

Taking into account the characteristics of the audience will allow an effective website to be created that will deliver the desired content to the target audience.

Compatibility and restrictions

Because of the market share of modern browsers (depending on your target market), the compatibility of your website with the viewers is restricted. For instance, a website that is designed for the majority of websurfers will be limited to the use of valid XHTML 1.0 Strict or older, Cascading Style Sheets Level 1, and 1024x768 display resolution. This is because Internet Explorer is not fully W3C standards compliant with the modularity of XHTML 1.1 and the majority of CSS beyond 1. A target market of more alternative browser (e.g. Firefox, Google Chrome, Safari and Opera) users allow for more W3C compliance and thus a greater range of options for a web designer.

Another restriction on webpage design is the use of different image file formats. The majority of users can support GIF, JPEG, and PNG (with restrictions). Again Internet Explorer is the major restriction here, not fully supporting PNG's advanced transparency features, resulting in the GIF format still being the most widely used graphic file format for transparent images.

Many website incompatibilities go unnoticed by the designer and unreported by the users. The only way to be certain a website will work on a particular platform is to test it on that platform.

Planning documentation

Documentation is used to visually plan the site while taking into account the purpose, audience and content, to design the site structure, content and interactions that are most suitable for the website. Documentation may be considered a prototype for the website – a model which allows the website layout to be reviewed, resulting in suggested changes, improvements and/or enhancements. This review process increases the likelihood of success of the website.

The first step may involve information architecture in which the content is categorized and the information structure is formulated. The information structure is used to develop a document or visual diagram called a site map. This creates a visual of how the web pages or content will be interconnected, and may help in deciding what content will be placed on what pages.

In addition to planning the structure, the layout and interface of individual pages may be planned using a storyboard. In the process of storyboarding, a record is made of the description, purpose and title of each page in the site, and they are linked together according to the most effective and logical diagram type. Depending on the number of pages required for the website, documentation methods may include using pieces of paper and drawing lines to connect them, or creating the storyboard using computer software.

Website design

Web design is different than traditional print publishing. Every website is an information display container, just as a book is a container; and every web page is like the page in a book. However the end size and shape of the web page is not known to the web designer, whereas the print designer will know exactly what size paper he will be printing on.

For the typical web sites, the basic aspects of design are:

- The *content:* the substance, and information on the site should be relevant to the site and should target the area of the public that the website is concerned with.
- The *usability:* the site should be user-friendly, with the interface and navigation simple and reliable.

- The *appearance:* the graphics and text should include a single style that flows throughout, to show consistency. The style should be professional, appealing and relevant.
- The *structure:* of the web site as a whole.

A web site typically consists of text, images, animation and /or video. The first page of a web site is known as the Home page or Index Page. Some web sites use what is commonly called a Splash Page. Splash pages might include a welcome message, language or region selection, or disclaimer, however search engines, in general, favor web sites that don't do this which has caused these types of pages to fall out of favor. Each web page within a web site is a file which has its own URL. After each web page is created, they are typically linked together using a navigation menu composed of hyperlinks.

Once a web site is completed, it must be published or uploaded in order to be viewable to the public over the internet. This may be done using an FTP client.

Multidisciplinary requirements

Web site design crosses multiple disciplines of multiple information systems, information technology, marketing, and communication design. The web site is an information system whose components are sometimes classified as front-end and back-end. The observable content (e.g. page layout, user interface, graphics, text, audio) is known as the front-end. The back-end comprises the organization and efficiency of the source code, invisible scripted functions, and the server-side components that process the output from the front-end. Depending on the size of a web development project, it may be carried out by a multi-skilled individual (sometimes called a web master), or a project manager may oversee collaborative design between group members with specialized skills.

Environment

Layout is a double edged sword: on the one hand, it is the expression of a framework that actively shapes the web designer. On the other hand, as the designer adapts that framework to projects, layout is the means of content delivery. Publishing a web engages communication **throughout** the production process as well as **within** the product created. Publication implies adaptation of culture and content standards. Web design incorporates multiple intersections between many layers of technical and social understanding, demanding creative direction, design element structure, and some form of social organization. Differing goals and methods resolve effectively in successful deployment of education, software and team management during the design process. However, many competing and evolving platforms and environments challenge acceptance, completion and continuity of every design product.

Collaboration

Early web design was less integrated with companies' advertising campaigns, customer transactions, extranets, intranets and social networking. Web sites were seen largely as static online brochures or database connection points, disconnected from the broader

scopes of a business or project. Many web sites are still disconnected from the broader project scope. As a result, many web sites are needlessly difficult to use, indirect in their way of communicating, and suffer from a 'disconnected' or ineffective bureaucratic information architecture.

Form versus function

A web developer may pay more attention to how a page looks while neglecting other copywriting and search engine optimization functions such as the readability of text, the ease of navigating the site, or how easily the visitors are going to find the site. As a result, the designers may end up in disputes where some want more decorative graphics at the expense of keyword-rich text, bullet lists, and text links. Assuming a false dichotomy that form and function are mutually exclusive overlooks the possibility of integrating multiple disciplines for a collaborative and synergistic solution. In many cases form follows function. Because some graphics serve communication purposes in addition to aesthetics, how well a site works may depend on the graphic design ideas as well as the professional writing considerations.

When using a lot of graphics, or sending a lot of instructions to the end client computer, a web page may load slowly, often irritating the user. This has become less of a problem as the internet has evolved with high-speed internet and the use of vector graphics. However there is still an ongoing engineering challenge to increase bandwidth and an artistic challenge to minimize the amount of graphics and their file sizes. This challenge is compounded since increased bandwidth encourages more graphics with larger file sizes.

Layout

Layout types

Layout refers to the dimensioning of content in a device display, and the delivery of media in a content related stream. Web design layouts result in visual content frameworks: these frameworks can be fixed, they can use units of measure that are relative, or they can provide fluid layout with proportional dimensions. The deployment flowchart (a useful tool on any design project) should address content layout. Many units of measure exist, but here are some popular dimension formats:

- Pixel measure results in fixed or static content
- Em measure results in proportional content that is relative to font-size
- Percent measure results in fluid content that shrinks and grows to "fit" display windows

Proportional, liquid and hybrid layout are also referred to as dynamic design. Hybrid layout incorporates any combination of fixed, proportional or fluid elements within (or pointing to) a single page. The hybrid web design framework is made possible by digital internet conventions generally prescribed by the W3C. If any layout does not appear as it should, it is very likely that it does not conform to standard design principles, or that those standards conflict with standard layout elements. Current knowledge of standards is essential to effective hybrid design.

The earliest web pages used fixed layouts without exception. In many business pages fixed layouts are preferred today as they easily contain static tabled information. Fixed layout enforces device display convention, as viewers must set their display to at least a certain width to easily view content. This width can include display of corporate logos, cautions, advertisements and any other target content. Design frameworks for fixed layout may need to include coding for multiple display devices.

Hybrid design maintains most static content control, but is adapted to textual publishing, and for readers, to conventional (printed) display. Hybrid layouts are generally easy on the eye and are found on most sites that distribute traditional images and text to readers. For some sites, hybrid design makes an otherwise cold text column appear warm and balanced. A good example of hybrid layout is Wordpress, where liquid design is now optional, and movie and auditory media is stretching the envelope.

Fluid design is useful where content is delivered to an 'unknown device' population. Appropriate liquid code displays images, text and spaces proportional to display size. Someone with a handheld can see view and interact with the same content as someone using a large desktop monitor. However, scaling of content for a variety of devices has more recently evolved with modern web browsers, allowing users to see the same layout across all devices.

Layout concerns

With the coming of numerous monitor sizes, "fluid" web sites are becoming less common. The result is that fluid layouts look "old" because they were typically used more in the early days of the internet. In dealing with font layout, even expressed as ems, a static core cannot be escaped and often anchors most page content. However, as new standards are adopted by device manufacturers, viewers notice a wider spectrum of content and a greater interaction between and through content. For the World Wide Web Consortium drawing up tomorrows layout conventions, new media types and methods are increasingly in the mix. It is a true double axiom that 'content is all about layout', and 'layout is all about content'. We could say that layout is what designers squeeze into available technology — content is the culture manifested in the layout. "Space' is the envelope holding layout and content together. Space communicates style (layout appearance) to the target population. Understanding how to adapt space to this layout-content relationship is essential to web design. Every design's survivability depends on its sensitivity to emerging technology (within the cultures that its framework is servicing), and immediate acceptance depends on the layout or presentation of that content. On every page, no content is more susceptible to changes and variations in standards, than space. While the professional designer casually admits that 90% of design code is used to adapt space, most of his current work deploys spatial manipulations being used to actively reshape Internet communication.

Conceptual barriers to adequate layout abound! Presently layout is challenged by conflicting convention that makes it impossible to fit liquid and hybrid layout to the bottom corners of a display. Simply put, display device manufactures use the top right and/or left corners to display content. For non-standard equipment, setting custom fixed layout to their device is still seen by some businesses as a means of increasing revenue,

as they can sell a 'unique' display. This business approach, domainating the digital market at the end of the last century, is not so useful today. However, some would claim a decade behind schedule, CSS3 and HTML5 are finally taking the four penultimate display reference points seriously.

A common misconception among designers is to assume their layout is liquid because initial space and text container widths are in percents. However, their 'liquid' framework, while adhering to focused conventions, failed to manage graphic content. A subsequent edit placing a large image on the page, destroys the page appearance. When managing a design framework, it is critical that layout address content, convention and user interaction.

Device

On the Web the designer has no control over several factors, including the size of the browser window, the web browser used, the input devices used (operating system, mouse, touch screen, voice command, text, teletype, cell phone, or other hand-held), and the size, design, and other characteristics of the fonts that users have available (installed) and enabled (preference) on their device. Unique manufacture and conflicting device contentions are further complicated by varying browser interpretations of the same content, and some content automatically can trigger browser changes. Web designers do well to study and become proficient at removing competitive device and software markup so that web pages display as they are coded to display. Eric Meyers, a well known educator and developer, is one of many resources who have spear-headed HTML reset coding. While they cannot yet leave one local environment to control another, web designers can adjust target environments to remove much common markup that alters or corrupts their web content. Because device manufacturers are highly protective of their patent markup, Meyers and others caution that reset remains *experimental*.

Tableless web design

When Netscape Navigator 4 dominated the browser market, the popular solution available for designers to lay out a web page was by using tables. Often even simple designs for a page would require dozens of tables nested in each other. Many web templates in Dreamweaver and other WYSIWYG editors still use this technique today. Navigator 4 didn't support CSS to a useful degree, so it simply wasn't used.

After the browser wars subsided, and the dominant browsers such as Internet Explorer became more W3C compliant, designers started turning toward CSS as an alternate means of laying out their pages. CSS proponents say that tables should be used only for tabular data, not for layout. Using CSS instead of tables also returns HTML to a semantic markup, which helps bots and search engines understand what's going on in a web page. All modern web browsers support CSS with different degrees of limitations.

However, one of the main points against CSS is that by relying on it exclusively, control is essentially relinquished as each browser has its own quirks which result in a slightly different page display. This is especially a problem as not every browser supports the same subset of CSS rules. There are the means to apply different styles depending on

which browser and version are used but incorporating these exceptions makes maintaining the style sheets more difficult as there are styles in more than one place to update.

For designers who are used to table-based layouts, developing web sites in CSS often becomes a matter of trying to replicate what can be done with tables, leading some to find CSS design rather cumbersome due to lack of familiarity. For example, at one time it was rather difficult to produce certain design elements, such as vertical positioning, and full-length footers in a design using absolute positions. With the abundance of CSS resources available online today, though, designing with reasonable adherence to standards involves little more than applying CSS 2.1 or CSS 3 to properly structured markup.

These days most modern browsers have solved most of these quirks in CSS rendering and this has made many different CSS layouts possible. However, some people continue to use old browsers, and designers need to keep this in mind, and allow for graceful degrading of pages in older browsers. Most notable among these old browsers is Internet Explorer 6, which is viewed in the web design community as becoming the new Netscape Navigator 4 — a block that holds the World Wide Web back from converting to CSS design. However, the W3 Consortium has made CSS in combination with XHTML the standard for web design.

Content Management

Many sites require frequent content changes and new content publishing at short notice. Content Management Systems (CMS) allow non-technical contributors to maintain and update site content without programing knowledge or special software tools. Typically content on the website is editable using a "What You See Is What You Get" (WYSIWYG) model. In addition to maintaining existing content, CMS administrators can upload images or videos, create pages, sections or categories, and add or edit menu structures. There are several options popular to current use.

Chapter 3

Web Page

A **web page** or **webpage** is a document or information resource that is suitable for the World Wide Web and can be accessed through a web browser and displayed on a monitor or mobile device. This information is usually in HTML or XHTML format, and may provide navigation to other web pages via hypertext links. Web pages frequently subsume other resources such as style sheets, scripts and images into their final presentation.

Web pages may be retrieved from a local computer or from a remote web server. The web server may restrict access only to a private network, e.g. a corporate intranet, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using Hypertext Transfer Protocol (HTTP).

Web pages may consist of files of static text and other content stored within the web server's file system (static web pages), or may be constructed by server-side software when they are requested (dynamic web pages). Client-side scripting can make web pages more responsive to user input once on the client browser.

Color, typography, illustration, and interaction

Web pages usually include information as to the colors of text and backgrounds and very often also contain links to images and sometimes other types of media to be included in the final view. Layout, typographic and color-scheme information is provided by Cascading Style Sheet (CSS) instructions, which can either be embedded in the HTML or can be provided by a separate file, which is referenced from within the HTML. The latter case is especially relevant where one lengthy stylesheet is relevant to a whole website: due to the way HTTP works, the browser will only download it once from the web server and use the cached copy for the whole site. Images are stored on the web server as separate files, but again HTTP allows for the fact that once a web page is downloaded to a browser, it is quite likely that related files such as images and stylesheets will be requested as it is processed. An HTTP 1.1 web server will maintain a connection with the browser until all related resources have been requested and provided. Web browsers usually render images along with the text and other material on the displayed web page.

Dynamic behavior

Client-side computer code such as JavaScript or code implementing Ajax techniques can be provided either embedded in the HTML of a web page or, like CSS stylesheets, as separate, linked downloads specified in the HTML. These scripts may run on the client computer, if the user allows.

Browsers

A web browser can have a Graphical User Interface, like Internet Explorer, Mozilla Firefox, Chrome and Opera, or can be text-based, like Lynx or Links.

Web users with disabilities often use assistive technologies and adaptive strategies to access web pages. Users may be color blind, may or may not want to use a mouse perhaps due to repetitive stress injury or motor-neurone problems, may be deaf and require audio to be captioned, may be blind and using a screen reader or braille display, may need screen magnification, etc.

Disabled and able-bodied users may disable the download and viewing of images and other media, to save time, network bandwidth or merely to simplify their browsing experience. Users of mobile devices often have restricted displays and bandwidth. Anyone may prefer not to use the fonts, font sizes, styles and color schemes selected by the web page designer and may apply their own CSS styling to the page.

The World Wide Web Consortium (W3C) and Web Accessibility Initiative (WAI) recommend that all web pages should be designed with all of these options in mind.

Elements

A *web page*, as an information set, can contain numerous types of information, which is able to be seen, heard or interact by the end user:

Perceived (rendered) information:

- *Textual information*: with diverse render variations.
- *Non-textual information*:
  - *Static images* may be raster graphics, typically GIF, JPEG or PNG; or vector formats such as SVG or Flash.
  - *Animated images* typically Animated GIF and SVG, but also may be Flash, Shockwave, or Java applet.
  - Audio, typically MP3, ogg or various proprietary formats.
  - Video, WMV (Windows), RM (Real Media), FLV (Flash Video), MPG, MOV (QuickTime)
- *Interactive information*
  - For "on page" interaction:
    - *Interactive text*

        - *Interactive illustrations*: ranging from "click to play" images to games, typically using *script orchestration*, Flash, Java applets, SVG, or Shockwave.
        - *Buttons*: forms providing alternative interface, typically for use with *script orchestration* and DHTML.
    - For "between pages" interaction:
        - *Hyperlinks*: standard "change page" reactivity.
        - *Forms*: providing more interaction with the server and server-side databases.

Internal (hidden) information:

- *Comments*
- *Linked Files through Hyperlink (Like DOC,XLS,PDF,etc).*
- *Metadata* with semantic meta-information, Charset information, Document Type Definition (DTD), etc.
- *Diagramation and style information*: information about rendered items (like image size attributes) and visual specifications, as Cascading Style Sheets (CSS).
- *Scripts*, usually JavaScript, complement interactivity and functionality.

Note: on server-side the web page may also have "Processing Instruction Information Items".

The web page can also contain dynamically adapted information elements, dependent upon the rendering browser or end-user location (through the use of IP address tracking and/or "cookie" information).

From a more general/wide point of view, some information (grouped) elements, like a navigation bar, are uniform for all website pages, like a standard. These kind of "website standard information" are supplied by technologies like web template systems.

Rendering

Web pages will often require more screen space than is available for a particular display resolution. Most modern browsers will place a scrollbar (a sliding tool at the side of the screen that allows the user to move the page up or down, or side-to-side) in the window to allow the user to see all content. Scrolling horizontally is less prevalent than vertical scrolling, not only because such pages often do not print properly, but because it inconveniences the user more so than vertical scrolling would (because lines are horizontal; scrolling back and forth for every line is much more inconvenient than scrolling after reading a whole screen; also most computer keyboards have page up and down keys, and many computer mice have vertical scroll wheels, but the horizontal scrolling equivalents are rare).

When web pages are stored in a common directory of a web server, they become a website. A website will typically contain a group of web pages that are linked together, or have some other coherent method of navigation. The most important web page to have on a website is the index page. Depending on the web server settings, this index page can

have many different names, but the most common is index.html. When a browser visits the homepage for a website, or any URL pointing to a directory rather than a specific file, the web server will serve the index page to the requesting browser. If no index page is defined in the configuration, or no such file exists on the server, either an error or directory listing will be served to the browser.

A web page can either be a single HTML file, or made up of several HTML files using frames or Server Side Includes (SSIs). Frames have been known to cause problems with web accessibility, copyright, navigation, printing and search engine rankings, and are now less often used than they were in the 1990s. Both frames and SSIs allow certain content which appears on many pages, such as page navigation or page headers, to be repeated without duplicating the HTML in many files. Frames and the W3C recommended alternative of 2000, the <object> tag, also allow some content to remain in one place while other content can be scrolled using conventional scrollbars. Modern CSS and JavaScript client-side techniques can also achieve all of these goals and more.

When creating a web page, it is important to ensure it conforms to the World Wide Web Consortium (W3C) standards for HTML, CSS, XML and other standards. The W3C standards are in place to ensure all browsers which conform to their standards can display identical content without any special consideration for proprietary rendering techniques. A properly coded web page is going to be accessible to many different browsers old and new alike, display resolutions, as well as those users with audio or visual impairments.

URL

Typically, web pages today are becoming more dynamic. A dynamic web page is one that is created server-side when it is requested, and then served to the end-user. These types of web pages typically do not have a permalink, or a static URL, associated with them. Today, this can be seen in many popular forums, online shopping. This practice is intended to reduce the amount of static pages in lieu of storing the relevant web page information in a database. Some search engines may have a hard time indexing a web page that is dynamic, so static web pages can be provided in those instances.

Viewing

In order to graphically display a web page, a web browser is needed. This is a type of software that can retrieve web pages from the Internet. Most current web browsers include the ability to view the source code. Viewing a web page in a text editor will also display the source code, not the visual product.

Creation

To create a web page, a text editor or a specialized HTML editor is needed. In order to upload the created web page to a web server, traditionally an FTP client is needed.

The design of a web page is highly personal. A design can be made according to one's own preference, or a premade web template can be used. Web templates let web page

designers edit the content of a web page without having to worry about the overall aesthetics. Many people publish their own web pages using products like Tripod, or Angelfire. These web publishing tools offer free page creation and hosting up to a certain size limit.

Saving

While one is viewing a web page, a copy of it is saved locally; this is what is being viewed. Depending on the browser settings, this copy may be deleted at any time, or stored indefinitely, sometimes without the user realizing it. Most GUI browsers provide options for saving a web page more permanently. These may include:

- Save the rendered text without formatting or images, with hyperlinks reduced to plain text
- Save the HTML as it was served — Overall structure preserved, but some links may be broken
- Save the HTML with relative links changed to absolute ones so that hyperlinks are preserved
- Save the entire web page — All images and other resources including stylesheets and scripts are downloaded and saved in a new folder alongside the HTML, with links to them altered to refer to the local copies. Other relative links changed to absolute
- Save the HTML as well as all images and other resources into a single MHTML file. This is supported by Internet Explorer and Opera. Other browsers may support this if a suitable plugin has been installed.

Most operating systems allow applications such as web browsers not only to print the currently viewed web page to a printer, but optionally to "print" to a file that can be viewed or printed later. Some web pages are designed, for example by use of CSS, so that hyperlinks, menus and other navigation items, which will be useless on paper, are rendered into print with this in mind. Sometimes, the destination addresses of hyperlinks may be shown explicitly, either within the body of the page or listed at the end of the printed version. Web page designers may specify in CSS that non-functional menus, navigational blocks and other items may simply be absent from the printed version.

Chapter 4

World Wide Web

World Wide Web

The Web's historic logo designed by Robert Cailliau

Inventor	Tim Berners-Lee
Launch year	1991
Company	CERN
Availability	Worldwide

The **World Wide Web** (W3), abbreviated as **WWW** and commonly known as **the Web**, is a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks. Using concepts from earlier hypertext systems, English engineer and computer scientist Sir Tim Berners-Lee, now the Director of the World Wide Web Consortium, wrote a proposal in March 1989 for what would eventually become the World Wide Web. At CERN in Geneva, Switzerland, Berners-Lee and Belgian computer scientist Robert Cailliau proposed in 1990 to use "HyperText ... to link and access information of various kinds as a web of nodes in which the user can browse at will", and publicly introduced the project in December.

"The World-Wide Web was developed to be a pool of human knowledge, and human culture, which would allow collaborators in remote sites to share their ideas and all aspects of a common project."

History

In the May 1970 issue of *Popular Science* magazine Arthur C. Clarke was reported to have predicted that satellites would one day "bring the accumulated knowledge of the world to your fingertips" using a console that would combine the functionality of the Xerox, telephone, television and a small computer, allowing data transfer and video conferencing around the globe.

In March 1989, Tim Berners-Lee wrote a proposal that referenced ENQUIRE, a database and software project he had built in 1980, and described a more elaborate information management system.

With help from Robert Cailliau, he published a more formal proposal (on November 12, 1990) to build a "Hypertext project" called "WorldWideWeb" (one word, also "W3") as a "web" of "hypertext documents" to be viewed by "browsers" using a client–server architecture. This proposal estimated that a read-only web would be developed within three months and that it would take six months to achieve "the creation of new links and new material by readers, [so that] authorship becomes universal" as well as "the automatic notification of a reader when new material of interest to him/her has become available." See Web 2.0 and RSS/Atom, which have taken a little longer to mature.

The proposal was modeled after the Dynatext SGML reader by Electronic Book Technology, a spin-off from the Institute for Research in Information and Scholarship at Brown University. The Dynatext system, licensed by CERN, was technically advanced and was a key player in the extension of SGML ISO 8879:1986 to Hypermedia within HyTime, but it was considered too expensive and had an inappropriate licensing policy for use in the general high energy physics community, namely a fee for each document and each document alteration.

This NeXT Computer used by Tim Berners-Lee at CERN became the first web server

The CERN datacenter in 2010 housing some www servers

A NeXT Computer was used by Berners-Lee as the world's first web server and also to write the first web browser, WorldWideWeb, in 1990. By Christmas 1990, Berners-Lee had built all the tools necessary for a working Web: the first web browser (which was a web editor as well); the first web server; and the first web pages, which described the project itself. On August 6, 1991, he posted a short summary of the World Wide Web project on the alt.hypertext newsgroup. This date also marked the debut of the Web as a publicly available service on the Internet. The first photo on the web was uploaded by Berners-Lee in 1992, an image of the CERN house band Les Horribles Cernettes.

Web as a "Side Effect" of the 40 years of Particle Physics Experiments. It happened many times during history of science that the most impressive results of large scale scientific efforts appeared far away from the main directions of those efforts... After the World War 2 the nuclear centers of almost all developed countries became the places with the highest concentration of talented scientists. For about four decades many of them were invited to the international CERN's Laboratories. So specific kind of the CERN's intellectual "entire culture" (as you called it) was constantly growing from one generation of the scientists and engineers to another. When the concentration of the human talents per square foot of the CERN's Labs reached the critical mass, it caused an intellectual explosion The Web -- crucial point of human's history -- was born... Nothing could be compared to it... We cant imagine yet the real scale of the recent shake, because there has not been so fast growing multi-dimension social-economic processes in human history...

The first server outside Europe was set up at SLAC to host the SPIRES-HEP database. Accounts differ substantially as to the date of this event. The World Wide Web Consortium says December 1992, whereas SLAC itself claims 1991. This is supported by a W3C document entitled *A Little History of the World Wide Web*.

The crucial underlying concept of hypertext originated with older projects from the 1960s, such as the Hypertext Editing System (HES) at Brown University, Ted Nelson's Project Xanadu, and Douglas Engelbart's oN-Line System (NLS). Both Nelson and Engelbart were in turn inspired by Vannevar Bush's microfilm-based "memex", which was described in the 1945 essay "As We May Think".

Berners-Lee's breakthrough was to marry hypertext to the Internet. In his book *Weaving The Web*, he explains that he had repeatedly suggested that a marriage between the two technologies was possible to members of *both* technical communities, but when no one took up his invitation, he finally tackled the project himself. In the process, he developed a system of globally unique identifiers for resources on the Web and elsewhere: the Universal Document Identifier (UDI), later known as Uniform Resource Locator (URL) and Uniform Resource Identifier (URI); the publishing language HyperText Markup Language (HTML); and the Hypertext Transfer Protocol (HTTP).

The World Wide Web had a number of differences from other hypertext systems that were then available. The Web required only unidirectional links rather than bidirectional ones. This made it possible for someone to link to another resource without action by the owner of that resource. It also significantly reduced the difficulty of implementing web servers and browsers (in comparison to earlier systems), but in turn presented the chronic problem of link rot. Unlike predecessors such as HyperCard, the World Wide Web was

non-proprietary, making it possible to develop servers and clients independently and to add extensions without licensing restrictions. On April 30, 1993, CERN announced that the World Wide Web would be free to anyone, with no fees due. Coming two months after the announcement that the server implementation of the Gopher protocol was no longer free to use, this produced a rapid shift away from Gopher and towards the Web. An early popular web browser was ViolaWWW, which was based upon HyperCard.

Scholars generally agree that a turning point for the World Wide Web began with the introduction of the Mosaic web browser in 1993, a graphical browser developed by a team at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign (NCSA-UIUC), led by Marc Andreessen. Funding for Mosaic came from the U.S. *High-Performance Computing and Communications Initiative*, a funding program initiated by the *High Performance Computing and Communication Act of 1991*, one of several computing developments initiated by U.S. Senator Al Gore. Prior to the release of Mosaic, graphics were not commonly mixed with text in web pages and the Web's popularity was less than older protocols in use over the Internet, such as Gopher and Wide Area Information Servers (WAIS). Mosaic's graphical user interface allowed the Web to become, by far, the most popular Internet protocol.

The World Wide Web Consortium (W3C) was founded by Tim Berners-Lee after he left the European Organization for Nuclear Research (CERN) in October, 1994. It was founded at the Massachusetts Institute of Technology Laboratory for Computer Science (MIT/LCS) with support from the Defense Advanced Research Projects Agency (DARPA), which had pioneered the Internet; a year later, a second site was founded at INRIA (a French national computer research lab) with support from the European Commission DG InfSo; and in 1996, a third continental site was created in Japan at Keio University. By the end of 1994, while the total number of websites was still minute compared to present standards, quite a number of notable websites were already active, many of which are the precursors or inspiration for today's most popular services.

Connected by the existing Internet, other websites were created around the world, adding international standards for domain names and HTML. Since then, Berners-Lee has played an active role in guiding the development of web standards (such as the markup languages in which web pages are composed), and in recent years has advocated his vision of a Semantic Web. The World Wide Web enabled the spread of information over the Internet through an easy-to-use and flexible format. It thus played an important role in popularizing use of the Internet. Although the two terms are sometimes conflated in popular use, *World Wide Web* is not synonymous with *Internet*. The Web is an application built on top of the Internet.

Function

The terms Internet and World Wide Web are often used in every-day speech without much distinction. However, the Internet and the World Wide Web are not one and the same. The Internet is a global system of interconnected computer networks. In contrast, the Web is one of the services that runs on the Internet. It is a collection of interconnected documents and other resources, linked by hyperlinks and URLs. In short, the Web is an application running on the Internet.

Viewing a web page on the World Wide Web normally begins either by typing the URL of the page into a web browser, or by following a hyperlink to that page or resource. The web browser then initiates a series of communication messages, behind the scenes, in order to fetch and display it.

First, the browser resolves the server-name portion of the URL into an Internet Protocol address using the global, distributed Internet database known as the Domain Name System (DNS); this lookup returns an IP address such as *208.80.152.2*. The browser then requests the resource by sending an HTTP request across the Internet to the computer at that particular address. It makes the request to a particular application port in the underlying Internet Protocol Suite so that the computer receiving the request can distinguish an HTTP request from other network protocols such as e-mail delivery; the HTTP protocol normally uses port 80.

The computer receiving the HTTP request delivers it to Web server software listening for requests on port 80. If the web server can fulfill the request it sends an HTTP response back to the browser indicating success, which can be as simple as

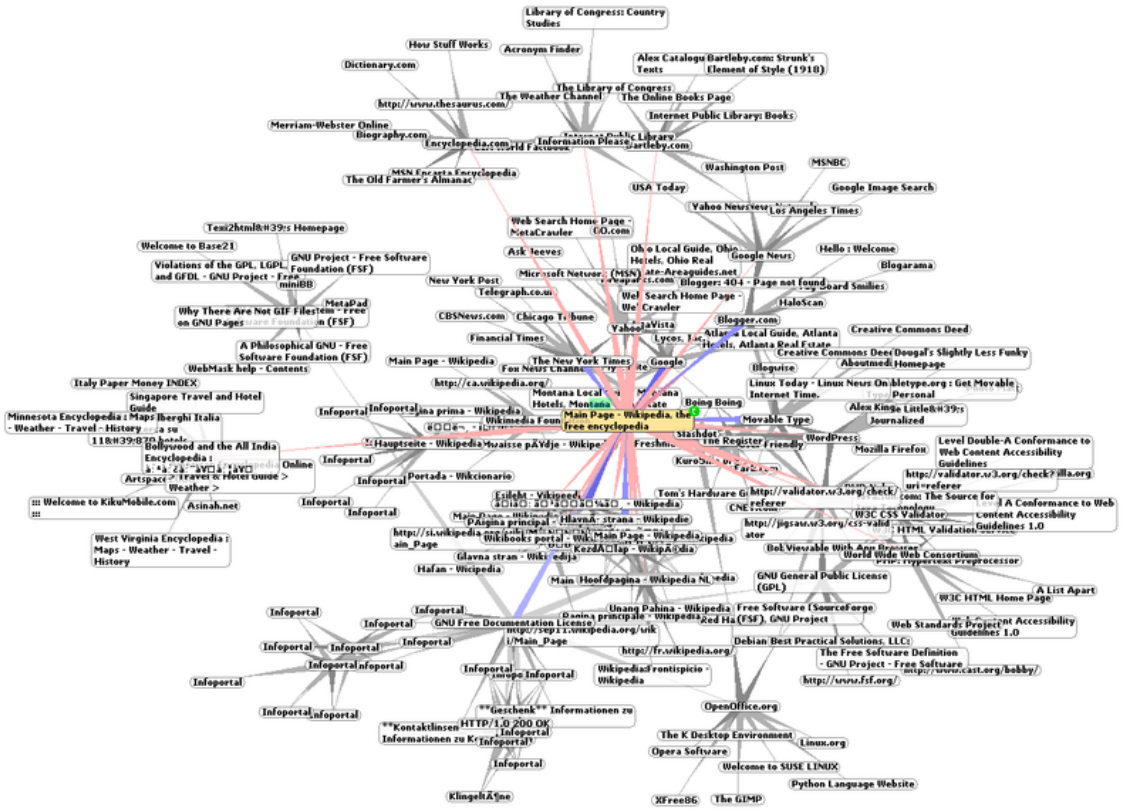
The web browser parses the HTML, interpreting the markup (<title>, <b> for bold, and such) that surrounds the words in order to draw that text on the screen.

Many web pages consist of more elaborate HTML which references the URLs of other resources such as images, other embedded media, scripts that affect page behavior, and Cascading Style Sheets that affect page layout. A browser that handles complex HTML will make additional HTTP requests to the web server for these other Internet media types. As it receives their content from the web server, the browser progressively renders the page onto the screen as specified by its HTML and these additional resources.

Linking

Most web pages contain hyperlinks to other related pages and perhaps to downloadable files, source documents, definitions and other web resources. In the underlying HTML, a hyperlink looks like

Early archive of the first Web site



Graphic representation of a minute fraction of the WWW, demonstrating hyperlinks

Such a collection of useful, related resources, interconnected via hypertext links is dubbed a *web* of information. Publication on the Internet created what Tim Berners-Lee first called the *WorldWideWeb* (in its original CamelCase, which was subsequently discarded) in November 1990.

Over time, many web resources pointed to by hyperlinks disappear, relocate, or are replaced with different content. This makes hyperlinks obsolete, a phenomenon referred to in some circles as link rot and the hyperlinks affected by it are often called dead links. The ephemeral nature of the Web has prompted many efforts to archive web sites. The Internet Archive, active since 1996, is one of the best-known efforts.

Dynamic updates of web pages

JavaScript is a scripting language that was initially developed in 1995 by Brendan Eich, then of Netscape, for use within web pages. The standardized version is ECMAScript. To overcome some of the limitations of the page-by-page model described above, some web applications also use Ajax (asynchronous JavaScript and XML). JavaScript is delivered with the page that can make additional HTTP requests to the server, either in response to user actions such as mouse-clicks, or based on lapsed time. The server's responses are used to modify the current page rather than creating a new page with each response. Thus the server only needs to provide limited, incremental information. Since multiple Ajax requests can be handled at the same time, users can interact with a page even while data

is being retrieved. Some web applications regularly poll the server to ask if new information is available.

WWW prefix

Many domain names used for the World Wide Web begin with *www* because of the long-standing practice of naming Internet hosts (servers) according to the services they provide. The hostname for a web server is often *www*, in the same way that it may be *ftp* for an FTP server, and *news* or *nntp* for a USENET news server. These host names appear as Domain Name System (DNS) subdomain names, as in www.example.com. The use of 'www' as a subdomain name is not required by any technical or policy standard; indeed, the first ever web server was called nxoc01.cern.ch, and many web sites exist without it. Many established websites still use 'www', or they invent other subdomain names such as 'www2', 'secure', etc. Many such web servers are set up such that both the domain root (e.g., example.com) and the *www* subdomain (e.g., www.example.com) refer to the same site; others require one form or the other, or they may map to different web sites.

The use of a subdomain name is useful for load balancing incoming web traffic by creating a CNAME record that points to a cluster of web servers. Since, currently, only a subdomain can be cname'ed the same result cannot be achieved by using the bare domain root.

When a user submits an incomplete website address to a web browser in its address bar input field, some web browsers automatically try adding the prefix "www" to the beginning of it and possibly ".com", ".org" and ".net" at the end, depending on what might be missing. For example, entering 'microsoft' may be transformed to *http://www.microsoft.com/* and 'openoffice' to *http://www.openoffice.org*. This feature started appearing in early versions of Mozilla Firefox, when it still had the working title 'Firebird' in early 2003, from a much more ancient practice in browsers such as Lynx. It is reported that Microsoft was granted a US patent for the same idea in 2008, but only for mobile devices.

The scheme specifiers (*http://* or *https://*) in URIs refer to the Hypertext Transfer Protocol and to HTTP Secure respectively and so define the communication protocol to be used for the request and response. The HTTP protocol is fundamental to the operation of the World Wide Web; the added encryption layer in HTTPS is essential when confidential information such as passwords or banking information are to be exchanged over the public Internet. Web browsers usually prepend the scheme to URLs too, if omitted.

In English, *www* is pronounced by individually pronouncing the name of characters (*double-u double-u double-u*). Although some technical users pronounce it *dub-dub-dub* this is not widespread. The English writer Douglas Adams once quipped in The Independent on Sunday (1999): "The World Wide Web is the only thing I know of whose shortened form takes three times longer to say than what it's short for," with Stephen Fry later pronouncing it in his "Podgrammes" series of podcasts as "wuh wuh wuh." In Mandarin Chinese, *World Wide Web* is commonly translated via a phono-semantic matching to *wàn wéi wăng* (万维网), which satisfies *www* and literally means "myriad

dimensional net", a translation that very appropriately reflects the design concept and proliferation of the World Wide Web. Tim Berners-Lee's web-space states that *World Wide Web* is officially spelled as three separate words, each capitalized, with no intervening hyphens.

Privacy

Computer users, who save time and money, and who gain conveniences and entertainment, may or may not have surrendered the right to privacy in exchange for using a number of technologies including the Web. Worldwide, more than a half billion people have used a social network service, and of Americans who grew up with the Web, half created an online profile and are part of a generational shift that could be changing norms. Facebook progressed from U.S. college students to a 70% non-U.S. audience, and in 2009 estimated that only 20% of its members use privacy settings. In 2010 (six years after co-founding the company), Mark Zuckerberg wrote, "we will add privacy controls that are much simpler to use".

Privacy representatives from 60 countries have resolved to ask for laws to complement industry self-regulation, for education for children and other minors who use the Web, and for default protections for users of social networks. They also believe data protection for personally identifiable information benefits business more than the sale of that information. Users can opt-in to features in browsers to clear their personal histories locally and block some cookies and advertising networks but they are still tracked in websites' server logs, and particularly web beacons. Berners-Lee and colleagues see hope in accountability and appropriate use achieved by extending the Web's architecture to policy awareness, perhaps with audit logging, reasoners and appliances.

In exchange for providing free content, vendors hire advertisers who spy on Web users and base their business model on tracking them. Since 2009, they buy and sell consumer data on exchanges (lacking a few details that could make it possible to de-anonymize, or identify an individual). Hundreds of millions of times per day, Lotame Solutions captures what users are typing in real time, and sends that text to OpenAmplify who then tries to determine, to quote a writer at *The Wall Street Journal*, "what topics are being discussed, how the author feels about those topics, and what the person is going to do about them".

Microsoft backed away in 2008 from its plans for strong privacy features in Internet Explorer, leaving its users (50% of the world's Web users) open to advertisers who may make assumptions about them based on only *one click* when they visit a website. Among services paid for by advertising, Yahoo! could collect the most data about users of commercial websites, about 2,500 bits of information per month about each typical user of its site and its affiliated advertising network sites. Yahoo! was followed by MySpace with about half that potential and then by AOL–TimeWarner, Google, Facebook, Microsoft, and eBay.

Security

The Web has become criminals' preferred pathway for spreading malware. Cybercrime carried out on the Web can include identity theft, fraud, espionage and intelligence gathering. Web-based vulnerabilities now outnumber traditional computer security concerns, and as measured by Google, about one in ten web pages may contain malicious code. Most Web-based attacks take place on legitimate websites, and most, as measured by Sophos, are hosted in the United States, China and Russia. The most common of all malware threats is SQL injection attacks against websites. Through HTML and URIs the Web was vulnerable to attacks like cross-site scripting (XSS) that came with the introduction of JavaScript and were exacerbated to some degree by Web 2.0 and Ajax web design that favors the use of scripts. Today by one estimate, 70% of all websites are open to XSS attacks on their users.

Proposed solutions vary to extremes. Large security vendors like McAfee already design governance and compliance suites to meet post-9/11 regulations, and some, like Finjan have recommended active real-time inspection of code and all content regardless of its source. Some have argued that for enterprise to see security as a business opportunity rather than a cost center, "ubiquitous, always-on digital rights management" enforced in the infrastructure by a handful of organizations must replace the hundreds of companies that today secure data and networks. Jonathan Zittrain has said users sharing responsibility for computing safety is far preferable to locking down the Internet.

Standards

Many formal standards and other technical specifications and software define the operation of different aspects of the World Wide Web, the Internet, and computer information exchange. Many of the documents are the work of the World Wide Web Consortium (W3C), headed by Berners-Lee, but some are produced by the Internet Engineering Task Force (IETF) and other organizations.

Usually, when web standards are discussed, the following publications are seen as foundational:

- Recommendations for markup languages, especially HTML and XHTML, from the W3C. These define the structure and interpretation of hypertext documents.
- Recommendations for stylesheets, especially CSS, from the W3C.
- Standards for ECMAScript (usually in the form of JavaScript), from Ecma International.
- Recommendations for the Document Object Model, from W3C.

Additional publications provide definitions of other essential technologies for the World Wide Web, including, but not limited to, the following:

- *Uniform Resource Identifier* (URI), which is a universal system for referencing resources on the Internet, such as hypertext documents and images. URIs, often called URLs, are defined by the IETF's RFC 3986 / STD 66: *Uniform Resource*

Identifier (URI): Generic Syntax, as well as its predecessors and numerous URI scheme-defining RFCs;
- *HyperText Transfer Protocol (HTTP)*, especially as defined by RFC 2616: *HTTP/1.1* and RFC 2617: *HTTP Authentication*, which specify how the browser and server authenticate each other.

Accessibility

Access to the Web is for everyone regardless of disability including visual, auditory, physical, speech, cognitive, or neurological. Accessibility features also help others with temporary disabilities like a broken arm or the aging population as their abilities change. The Web is used for receiving information as well as providing information and interacting with society, making it essential that the Web be accessible in order to provide equal access and equal opportunity to people with disabilities. Tim Berners-Lee once noted, "The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect." Many countries regulate web accessibility as a requirement for websites. International cooperation in the W3C Web Accessibility Initiative led to simple guidelines that web content authors as well as software developers can use to make the Web accessible to persons who may or may not be using assistive technology.

Internationalization

The W3C Internationalization Activity assures that web technology will work in all languages, scripts, and cultures. Beginning in 2004 or 2005, Unicode gained ground and eventually in December 2007 surpassed both ASCII and Western European as the Web's most frequently used character encoding. Originally RFC 3986 allowed resources to be identified by URI in a subset of US-ASCII. RFC 3987 allows more characters—any character in the Universal Character Set—and now a resource can be identified by IRI in any language.

Statistics

Between 1995 and 2010, the number of Web users doubled, and was expected to surpass two billion in 2010.. According to a 2001 study, there were a massive over 550 billion documents on the Web, mostly in the invisible Web, or deep Web. A 2002 survey of 2,024 million Web pages determined that by far the most Web content was in English: 56.4%; next were pages in German (7.7%), French (5.6%), and Japanese (4.9%). A more recent study, which used Web searches in 75 different languages to sample the Web, determined that there were over 11.5 billion Web pages in the publicly indexable Web as of the end of January 2005. As of March 2009, the indexable web contains at least 25.21 billion pages. On July 25, 2008, Google software engineers Jesse Alpert and Nissan Hajaj announced that Google Search had discovered one trillion unique URLs. As of May 2009, over 109.5 million websites operated. Of these 74% were commercial or other sites operating in the .com generic top-level domain.

Statistics measuring a website's popularity are usually based either on the number of page views or associated server 'hits' (file requests) that it receives.

Speed issues

Frustration over congestion issues in the Internet infrastructure and the high latency that results in slow browsing has led to a pejorative name for the World Wide Web: the *World Wide Wait*. Speeding up the Internet is an ongoing discussion over the use of peering and QoS technologies. Other solutions to reduce the congestion can be found at W3C. Guidelines for Web response times are:

- 0.1 second (one tenth of a second). Ideal response time. The user doesn't sense any interruption.
- 1 second. Highest acceptable response time. Download times above 1 second interrupt the user experience.
- 10 seconds. Unacceptable response time. The user experience is interrupted and the user is likely to leave the site or system.

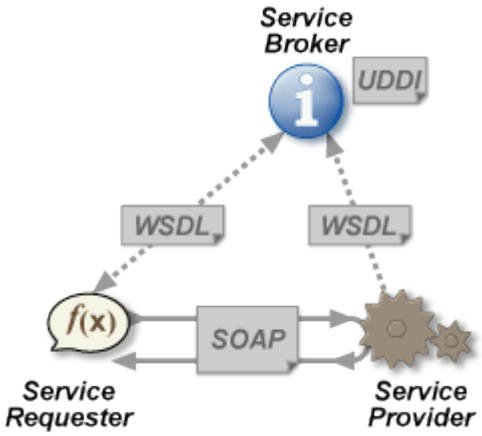
Caching

If a user revisits a Web page after only a short interval, the page data may not need to be re-obtained from the source Web server. Almost all web browsers cache recently obtained data, usually on the local hard drive. HTTP requests sent by a browser will usually only ask for data that has changed since the last download. If the locally cached data are still current, it will be reused. Caching helps reduce the amount of Web traffic on the Internet. The decision about expiration is made independently for each downloaded file, whether image, stylesheet, JavaScript, HTML, or whatever other content the site may provide. Thus even on sites with highly dynamic content, many of the basic resources only need to be refreshed occasionally. Web site designers find it worthwhile to collate resources such as CSS data and JavaScript into a few site-wide files so that they can be cached efficiently. This helps reduce page download times and lowers demands on the Web server.

There are other components of the Internet that can cache Web content. Corporate and academic firewalls often cache Web resources requested by one user for the benefit of all. Some search engines also store cached content from websites. Apart from the facilities built into Web servers that can determine when files have been updated and so need to be re-sent, designers of dynamically generated Web pages can control the HTTP headers sent back to requesting users, so that transient or sensitive pages are not cached. Internet banking and news sites frequently use this facility. Data requested with an HTTP 'GET' is likely to be cached if other conditions are met; data obtained in response to a 'POST' is assumed to depend on the data that was POSTed and so is not cached.

Chapter 5

Web Service



Web services architecture.

A **Web service** is a method of communication between two electronic devices.

The W3C defines a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."

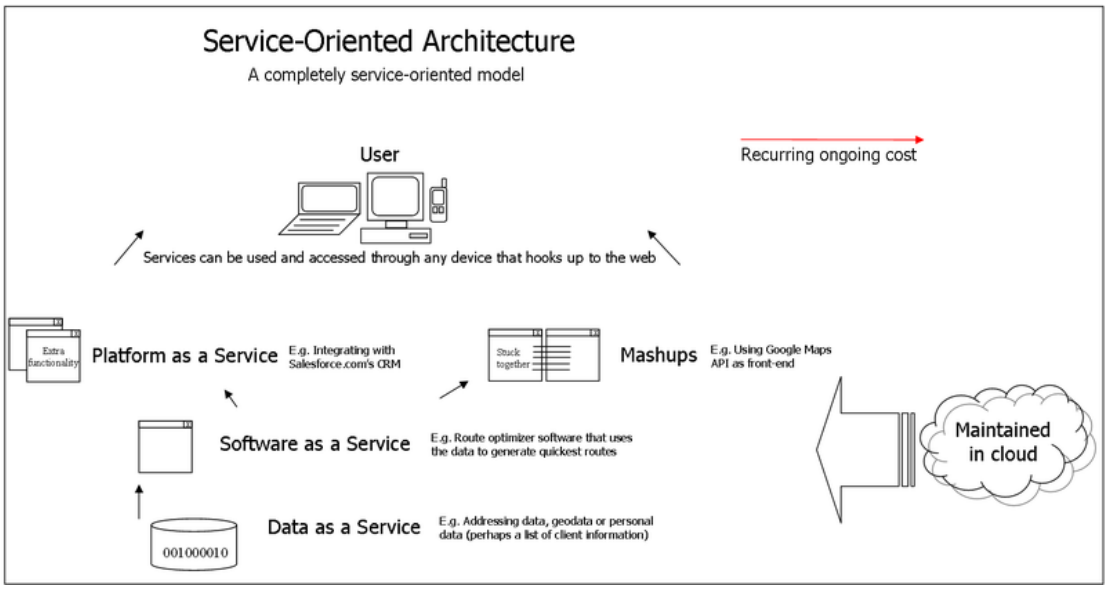
The W3C also states, "We can identify two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations."

Big Web services

"Big Web services" use Extensible Markup Language (XML) messages that follow the SOAP standard and have been popular with traditional enterprise. In such systems, there is often a machine-readable description of the operations offered by the service written in the Web Services Description Language (WSDL). The latter is not a requirement of a SOAP *endpoint*, but it is a prerequisite for automated client-side code generation in many Java and .NET SOAP frameworks (frameworks such as Spring, Apache Axis2 and

Apache CXF being notable exceptions). Some industry organizations, such as the WS-I, mandate both SOAP and WSDL in their definition of a Web service.

Web API



Web services in a service-oriented architecture.

Web API is a development in Web services (in a movement called Web 2.0) where emphasis has been moving away from SOAP based services towards Representational State Transfer (REST) based communications. REST services do not require XML, SOAP, or WSDL service-API definitions.

Web APIs allow the combination of multiple Web services into new applications known as mashups.

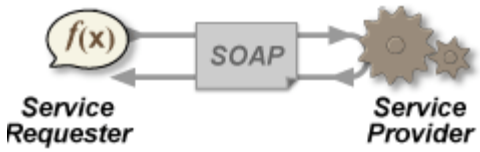
When used in the context of Web development, Web API is typically a defined set of Hypertext Transfer Protocol (HTTP) request messages along with a definition of the structure of response messages, usually expressed in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

When running composite Web services, each sub service can be considered autonomous. The user has no control over these services. Also the Web services themselves are not reliable; the service provider may remove, change or update their services without giving notice to users. The reliability and fault tolerance is not well supported; faults may happen during the execution. Exception handling in the context of Web services is still an open research issue. Still it can be handled by responding with an error object to the client.

Styles of use

Web services are a set of tools that can be used in a number of ways. The three most common styles of use are RPC, SOA and REST.

Remote procedure calls



Architectural elements involved in the XML-RPC.

RPC Web services present a distributed function (or method) call interface that is familiar to many developers. Typically, the basic unit of RPC Web services is the WSDL operation.

The first Web services tools were focused on RPC, and as a result this style is widely deployed and supported. However, it is sometimes criticized for not being loosely coupled, because it was often implemented by mapping services directly to language-specific functions or method calls. Many vendors felt this approach to be a dead end, and pushed for RPC to be disallowed in the WS-I Basic Profile.

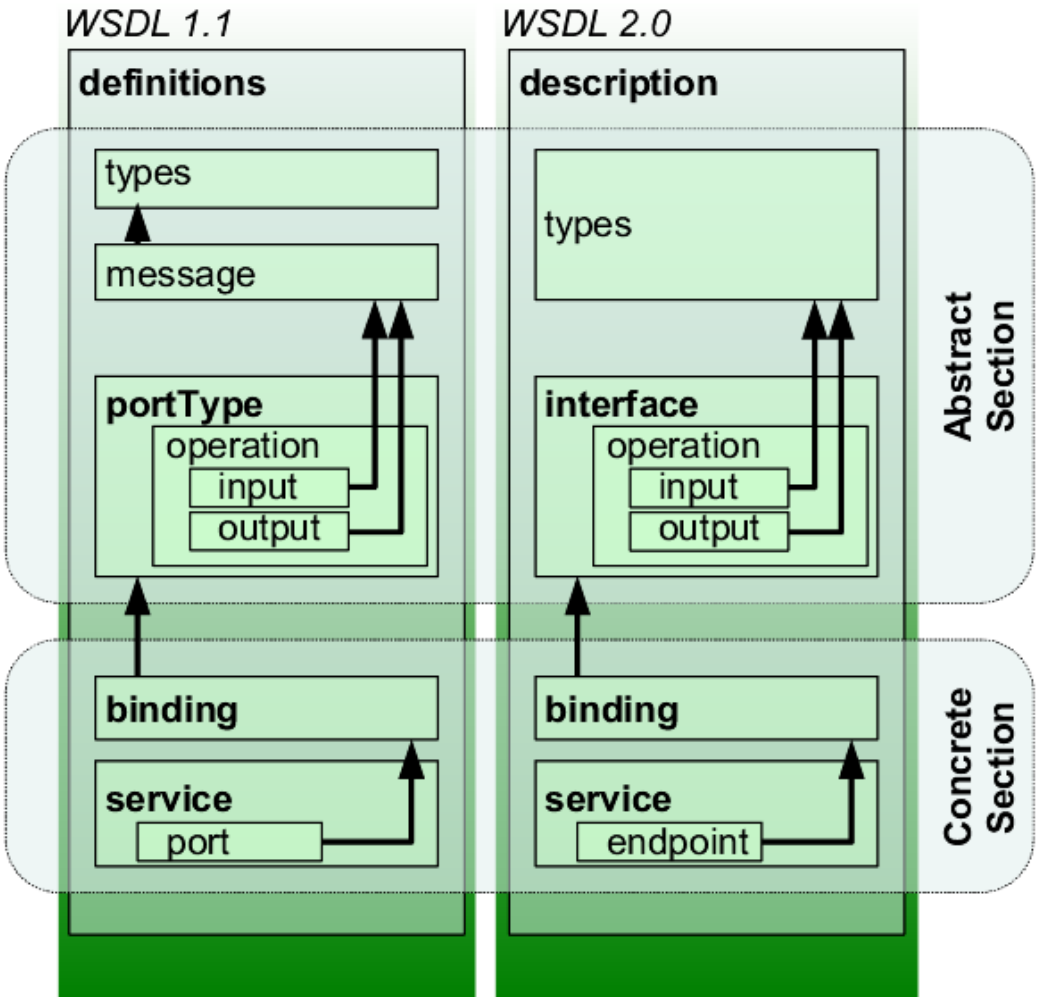
Other approaches with nearly the same functionality as RPC are Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) or Sun Microsystems's Java/Remote Method Invocation (RMI).

Service-oriented architecture

Web services can also be used to implement an architecture according to service-oriented architecture (SOA) concepts, where the basic unit of communication is a message, rather than an operation. This is often referred to as "message-oriented" services.

SOA Web services are supported by most major software vendors and industry analysts. Unlike RPC Web services, loose coupling is more likely, because the focus is on the "contract" that WSDL provides, rather than the underlying implementation details.

Middleware analysts use enterprise service buses that combine message-oriented processing and Web services to create an event-driven SOA. One example of an open-source ESB is Mule, another one is Open ESB.



Representation of concepts defined by WSDL 1.1 and WSDL 2.0 documents.

Representational state transfer (REST)

REST attempts to describe architectures that use HTTP or similar protocols by constraining the interface to a set of well-known, standard operations (like GET, POST, PUT, DELETE for HTTP). Here, the focus is on interacting with stateful resources, rather than messages or operations.

An architecture based on REST (one that is 'RESTful') can use WSDL to describe SOAP messaging over HTTP, can be implemented as an abstraction purely on top of SOAP (e.g., WS-Transfer), or can be created without using SOAP at all.

WSDL version 2.0 offers support for binding to all the HTTP request methods (not only GET and POST as in version 1.1) so it enables a better implementation of RESTful Web services. However, support for this specification is still poor in software development kits, which often offer tools only for WSDL 1.1.

Automated design methodologies

Automated tools can aid in the creation of a Web service. For services using WSDL it is possible to either automatically generate WSDL for existing classes (a bottom-up strategy) or to generate a class skeleton given existing WSDL (a top-down strategy).

- A developer using a bottom up method writes implementing classes first (in some programming language), and then uses a WSDL generating tool to expose methods from these classes as a Web service . This is often the simpler approach.
- A developer using a top down method writes the WSDL document first and then uses a code generating tool to produce the class skeleton, to be completed as necessary. This way is generally considered more difficult but can produce cleaner designs

Criticisms

Critics of non-RESTful Web services often complain that they are too complex and based upon large software vendors or integrators, rather than typical open source implementations. There are open source implementations like Apache Axis and Apache CXF.

One key concern of the REST Web service developers is that the SOAP WS toolkits make it easy to define new interfaces for remote interaction, often relying on introspection to extract the WSDL, since a minor change on the server (even an upgrade of the SOAP stack) can result in different WSDL and a different service interface. The client-side classes that can be generated from WSDL and XSD descriptions of the service are often similarly tied to a particular version of the SOAP endpoint and can break, if the endpoint changes or the client-side SOAP stack is upgraded. Well-designed SOAP endpoints (with handwritten XSD and WSDL) do not suffer from this but there is still the problem that a custom interface for every service requires a custom client for every service.

There are also concerns about performance due to Web services' use of XML as a message format and SOAP/HTTP in enveloping and transport, such as that published by the University of Wollongong in 2005 by N.A.B.Gray.

Chapter 6

Semantic Web

W3C's Semantic Web logo

The **Semantic Web** is a "web of data" that enables machines to understand the semantics, or meaning, of information on the World Wide Web. It extends the network of hyperlinked human-readable web pages by inserting machine-readable metadata about pages and how they are related to each other, enabling automated agents to access the Web more intelligently and perform tasks on behalf of users. The term was coined by Tim Berners-Lee, the inventor of the World Wide Web and director of the World Wide Web Consortium, which oversees the development of proposed Semantic Web standards. He defines the Semantic Web as "a web of data that can be processed directly and indirectly by machines."

The term "Semantic Web" is often used more specifically to refer to the formats and technologies that enable it. These technologies include the Resource Description Framework (RDF), a variety of data interchange formats (e.g. RDF/XML, N3, Turtle, N-Triples), and notations such as RDF Schema (RDFS) and the Web Ontology Language (OWL), all of which are intended to provide a formal description of concepts, terms, and relationships within a given knowledge domain.

Many of the technologies proposed by the W3C already exist and are used in various contexts, particularly those dealing with information that encompasses a limited and defined domain, and where sharing data is a common necessity, such as scientific research or data exchange among businesses. In addition, other technologies with similar goals have emerged, such as microformats. However, the Semantic Web as originally envisioned, a system that enables machines to understand and respond to complex human requests based on their meaning, has remained largely unrealized and its critics have questioned its feasibility.

Purpose

The main purpose of the **Semantic Web** is driving the evolution of the current Web by allowing users to use it to its full potential, thus allowing them to find, share, and combine information more easily. Humans are capable of using the Web to carry out tasks such as finding the Irish word for "folder," reserving a library book, and searching for a low price for a DVD. However, machines cannot accomplish all of these tasks without human direction, because web pages are designed to be read by people, not machines. The semantic web is a vision of information that can be interpreted by machines, so machines can perform more of the tedious work involved in finding, combining, and acting upon information on the web.

Tim Berners-Lee originally expressed the vision of the semantic web as follows:

I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize.

– Tim Berners-Lee, 1999

Semantic Web application areas are experiencing intensified interest due to the rapid growth in the use of the Web, together with the innovation and renovation of information content technologies. The Semantic Web is regarded as an integrator across different content, information applications and systems, it also provides mechanisms for the realisation of Enterprise Information Systems. The rapidity of the growth experienced provides the impetus for researchers to focus on the creation and dissemination of innovative Semantic Web technologies, where the envisaged 'Semantic Web' is long overdue. Often the terms 'Semantics', 'metadata', 'ontologies' and 'Semantic Web' are used inconsistently. In particular, these terms are used as everyday terminology by researchers and practitioners, spanning a vast landscape of different fields, technologies, concepts and application areas. Furthermore, there is confusion with regards to the current status of the enabling technologies envisioned to realise the Semantic Web. In a paper presented by Gerber, Barnard and Van der Merwe the Semantic Web landscape is charted and a brief summary of related terms and enabling technologies is presented. The architectural model proposed by Tim Berners-Lee is used as basis to present a status model that reflects current and emerging technologies.

Semantic Publishing

Semantic publishing will greatly benefit from the semantic web. In particular, the semantic web is expected to revolutionize scientific publishing, such as real-time publishing and sharing of experimental data on the Internet. This simple but radical idea is now being explored by W3C HCLS group's Scientific Publishing Task Force.

Semantic Blogging

Semantic blogging, like semantic publishing, will change the way blogs are read. Currently "the process of blogging inherently emphasizes metadata creation more than traditional Web publishing methodologies". Some blog users already tag their entries with topics, allowing for easier migration into a semantic web environment. It is intentionally saved in not only a human-readable format, but also in a machine-readable format as the tags can be linked easily to other blogs containing similar information. When a release of a game or movie occurs, bloggers tend to rate them using their own system. If there were to be a unified system, these blogs could easily become assimilated using similar semantics and give a user a score when searching using a semantic search. RSS feeds are another way that blogs already have machine-readable data that is easily accessible by the semantic web.

Web 3.0

Tim Berners-Lee has described the semantic web as a component of 'Web 3.0'.

The internet community as a whole tends to find the two terms "Semantic Web" and "Web 3.0" to be at least synonymous in concept if not completely interchangeable. The definition continues to vary depending on to whom you speak. The overwhelming consensus is that Web 3.0 is most assuredly the "next big thing" but there only lies speculation as to just what that might be. It will be an improvement in the respect that it will still contain Web 2.0 properties while continuing to add to its ever expanding lexicon and library of applications. There are some who claim that Web 3.0 will be more application based and center its efforts towards more graphically capable environments, "non-browser applications and non-computer based devices...geographic or location-based information retrieval" and even more applicable use and growth of Artificial Intelligence. For example, Conrad Wolfram, has argued that Web 3.0 is where "the computer is generating new information", rather than humans.

Others simply state their belief that Web 3.0 will primarily focus on dramatically improving the functionality and usability of search engines. An important factor that users must continue to keep in mind is that the transition to Web 2.0 from "Web 1.0" took approximately ten years. Given the same time frame, this next transition will not be complete until around the year 2015.

People keep asking what Web 3.0 is. I think maybe when you've got an overlay of scalable vector graphics – everything rippling and folding and looking misty — on Web 2.0 and access to a semantic Web integrated across a huge space of data, you'll have access to an unbelievable data resource..."

– Tim Berners-Lee, 2006

Highly specialized information silos, moderated by a cult of personality, validated by the community, and put into context with the inclusion of meta-data through widgets.

– Steve Spalding, 2007

Relationship to the hypertext web

Limitations of HTML

Many files on a typical computer can be loosely divided into documents and data. Documents like mail messages, reports, and brochures are read by humans. Data, like calendars, addressbooks, playlists, and spreadsheets are presented using an application program which lets them be viewed, searched and combined in many ways.

Currently, the World Wide Web is based mainly on documents written in Hypertext Markup Language (HTML), a markup convention that is used for coding a body of text interspersed with multimedia objects such as images and interactive forms. Metadata tags, for example

```
<meta name="keywords" content="computing, computer studies, computer">
<meta name="description" content="Cheap widgets for sale">
<meta name="author" content="John Doe">
```

provide a method by which computers can categorise the content of web pages.

With HTML and a tool to render it (perhaps web browser software, perhaps another user agent), one can create and present a page that lists items for sale. The HTML of this catalog page can make simple, document-level assertions such as "this document's title is 'Widget Superstore'", but there is no capability within the HTML itself to assert unambiguously that, for example, item number X586172 is an Acme Gizmo with a retail price of €199, or that it is a consumer product. Rather, HTML can only say that the span of text "X586172" is something that should be positioned near "Acme Gizmo" and "€199", etc. There is no way to say "this is a catalog" or even to establish that "Acme Gizmo" is a kind of title or that "€199" is a price. There is also no way to express that these pieces of information are bound together in describing a discrete item, distinct from other items perhaps listed on the page.

Semantic HTML refers to the traditional HTML practice of markup following intention, rather than specifying layout details directly. For example, the use of denoting "emphasis" rather than <i>, which specifies italics. Layout details are left up to the browser, in combination with Cascading Style Sheets. But this practice falls short of specifying the semantics of objects such as items for sale or prices.

Microformats represent unofficial attempts to extend HTML syntax to create machine-readable semantic markup about objects such as retail stores and items for sale.

Semantic Web solutions

The Semantic Web takes the solution further. It involves publishing in languages specifically designed for data: Resource Description Framework (RDF), Web Ontology Language (OWL), and Extensible Markup Language (XML). HTML describes documents and the links between them. RDF, OWL, and XML, by contrast, can describe arbitrary things such as people, meetings, or airplane parts. Tim Berners-Lee calls the

resulting network of Linked Data the Giant Global Graph, in contrast to the HTML-based World Wide Web.

These technologies are combined in order to provide descriptions that supplement or replace the content of Web documents. Thus, content may manifest itself as descriptive data stored in Web-accessible databases, or as markup within documents (particularly, in Extensible HTML (XHTML) interspersed with XML, or, more often, purely in XML, with layout or rendering cues stored separately). The machine-readable descriptions enable content managers to add meaning to the content, i.e., to describe the structure of the knowledge we have about that content. In this way, a machine can process knowledge itself, instead of text, using processes similar to human deductive reasoning and inference, thereby obtaining more meaningful results and helping computers to perform automated information gathering and research.

An example of a tag that would be used in a non-semantic web page:

```
<item>cat</item>
```

Encoding similar information in a semantic web page might look like this:

```
<item rdf:about="http://dbpedia.org/resource/Cat">Cat</item>
```

Skeptical reactions

Practical feasibility

Critics (e.g. Which Semantic Web?) question the basic feasibility of a complete or even partial fulfillment of the semantic web. Cory Doctorow's critique ("metacrap") is from the perspective of human behavior and personal preferences. For example, people lie: they may include spurious metadata into Web pages in an attempt to mislead Semantic Web engines that naively assume the metadata's veracity. This phenomenon was well-known with metatags that fooled the AltaVista ranking algorithm into elevating the ranking of certain Web pages: the Google indexing engine specifically looks for such attempts at manipulation. Peter Gärdenfors and Timo Honkela point out that logic-based semantic web technologies cover only a fraction of the relevant phenomena related to semantics.

Where semantic web technologies have found a greater degree of practical adoption, it has tended to be among core specialized communities and organizations for intra-company projects. The practical constraints toward adoption have appeared less challenging where domain and scope is more limited than that of the general public and the World-Wide Web.

The potential of an idea in fast progress

The original 2001 Scientific American article by Berners-Lee described an expected evolution of the existing Web to a Semantic Web. A complete evolution as described by Berners-Lee has yet to occur. In 2006, Berners-Lee and colleagues stated that: "This simple idea, however, remains largely unrealized." While the idea is still in the making, it

seems to evolve quickly and inspire many. Between 2007–2010 several scholars have already explored first applications and the social potential of the semantic web in the business and health sectors, and for social networking and even for the broader evolution of democracy, specifically, how a society forms its common will in a democratic manner through a semantic web

Censorship and privacy

Enthusiasm about the semantic web could be tempered by concerns regarding censorship and privacy. For instance, text-analyzing techniques can now be easily bypassed by using other words, metaphors for instance, or by using images in place of words. An advanced implementation of the semantic web would make it much easier for governments to control the viewing and creation of online information, as this information would be much easier for an automated content-blocking machine to understand. In addition, the issue has also been raised that, with the use of FOAF files and geo location meta-data, there would be very little anonymity associated with the authorship of articles on things such as a personal blog. Some of these concerns were addressed in the "Policy Aware Web" project and is an active research and development topic.

Doubling output formats

Another criticism of the semantic web is that it would be much more time-consuming to create and publish content because there would need to be two formats for one piece of data: one for human viewing and one for machines. However, many web applications in development are addressing this issue by creating a machine-readable format upon the publishing of data or the request of a machine for such data. The development of microformats has been one reaction to this kind of criticism. Another argument in defense of the feasibility of semantic web is the likely falling price of human intelligence tasks in digital labor markets like the Amazon Mechanical Turk.

Specifications such as eRDF and RDFa allow arbitrary RDF data to be embedded in HTML pages. The GRDDL (Gleaning Resource Descriptions from Dialects of Language) mechanism allows existing material (including microformats) to be automatically interpreted as RDF, so publishers only need to use a single format, such as HTML.

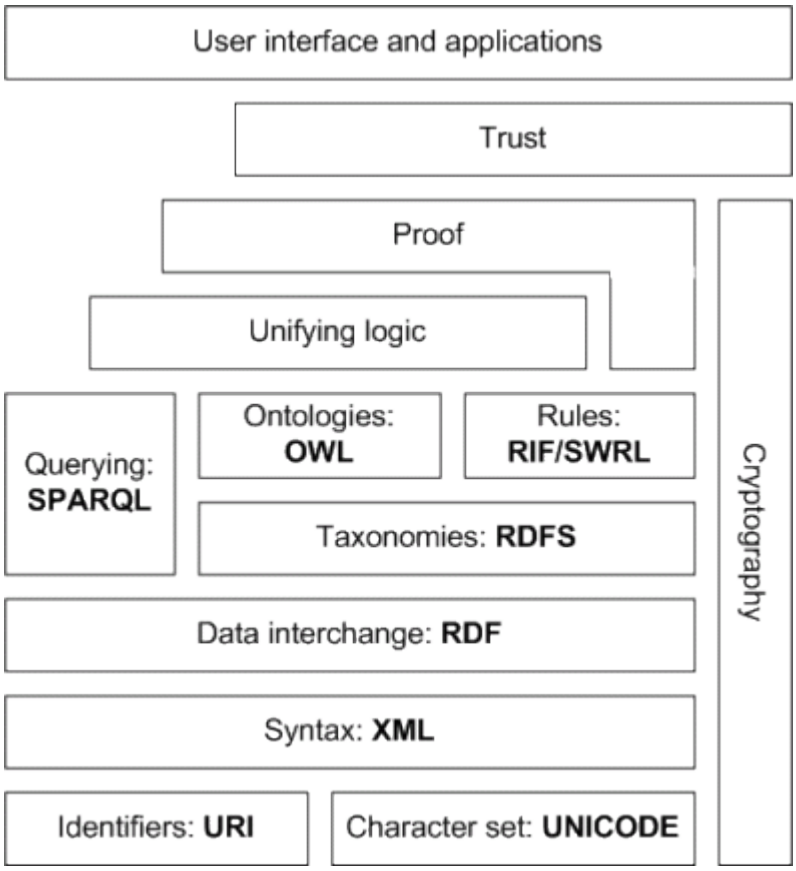
Need

The idea of a *semantic web*, able to describe and associate meaning with data necessarily involves more than simple XHTML mark-up code. It is based on an assumption that in order for it to be possible to endow machines with an ability to accurately interpret web homed content, far more than the mere ordered relationships involving letters and words, is necessary as underlying infrastructure (attendant to semantic issues). Otherwise, most of the supportive functionality would have been available in Web 2.0 (and before) and it would have been possible to derive a semantically capable Web with minor, incremental additions.

Additions to the infrastructure to support semantic functionality include latent dynamic network models that can, under certain conditions, be 'trained' to appropriately 'learn'

meaning based on order data, in the process 'learning' relationships with order (a kind of rudimentary working grammar).

Components



The Semantic Web Stack.

The semantic web comprises the standards and tools of XML, XML Schema, RDF, RDF Schema and OWL that are organized in the Semantic Web Stack. The OWL Web Ontology Language Overview describes the function and relationship of each of these components of the semantic web:

- XML provides an elemental syntax for content structure within documents, yet associates no semantics with the meaning of the content contained within. XML is not at present a necessary component of Semantic Web technologies in most cases, as alternative syntaxes exists, such as Turtle. Turtle is a de facto standard, but has not been through a formal standardization process.
- XML Schema is a language for providing and restricting the structure and content of elements contained within XML documents.
- RDF is a simple language for expressing data models, which refer to objects ("resources") and their relationships. An RDF-based model can be represented in XML syntax.

- RDF Schema extends RDF and is a vocabulary for describing properties and classes of RDF-based resources, with semantics for generalized-hierarchies of such properties and classes.
- OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.
- SPARQL is a protocol and query language for semantic web data sources.

Current ongoing standardizations include:

- Rule Interchange Format (RIF) as the Rule Layer of the Semantic Web Stack

Not yet fully realized layers include:

- Unifying Logic and Proof layers are undergoing active research.

The intent is to enhance the usability and usefulness of the Web and its interconnected resources through:

- Servers which expose existing data systems using the RDF and SPARQL standards. Many converters to RDF exist from different applications. Relational databases are an important source. The semantic web server attaches to the existing system without affecting its operation.
- Documents "marked up" with semantic information (an extension of the HTML <meta> tags used in today's Web pages to supply information for Web search engines using web crawlers). This could be machine-understandable information about the human-understandable content of the document (such as the creator, title, description, etc., of the document) or it could be purely metadata representing a set of facts (such as resources and services elsewhere in the site). (Note that *anything* that can be identified with a *Uniform Resource Identifier* (URI) can be described, so the semantic web can reason about animals, people, places, ideas, etc.) Semantic markup is often generated automatically, rather than manually.
- Common metadata vocabularies (ontologies) and maps between vocabularies that allow document creators to know how to mark up their documents so that agents can use the information in the supplied metadata (so that *Author* in the sense of 'the Author of the page' won't be confused with *Author* in the sense of a book that is the subject of a book review).
- Automated agents to perform tasks for users of the semantic web using this data
- Web-based services (often with agents of their own) to supply information specifically to agents (for example, a Trust service that an agent could ask if some online store has a history of poor service or spamming)

Challenges

Some of the challenges for the Semantic Web include vastness, vagueness, uncertainty, inconsistency, and deceit. Automated reasoning systems will have to deal with all of these issues in order to deliver on the promise of the Semantic Web.

- Vastness: The World Wide Web contains at least 24 billion pages as of this writing (June 13, 2010). The SNOMED CT medical terminology ontology contains 370,000 class names, and existing technology has not yet been able to eliminate all semantically duplicated terms. Any automated reasoning system will have to deal with truly huge inputs.
- Vagueness: These are imprecise concepts like "young" or "tall". This arises from the vagueness of user queries, of concepts represented by content providers, of matching query terms to provider terms and of trying to combine different knowledge bases with overlapping but subtly different concepts. Fuzzy logic is the most common technique for dealing with vagueness.
- Uncertainty: These are precise concepts with uncertain values. For example, a patient might present a set of symptoms which correspond to a number of different distinct diagnoses each with a different probability. Probabilistic reasoning techniques are generally employed to address uncertainty.
- Inconsistency: These are logical contradictions which will inevitably arise during the development of large ontologies, and when ontologies from separate sources are combined. Deductive reasoning fails catastrophically when faced with inconsistency, because "anything follows from a contradiction". Defeasible reasoning and paraconsistent reasoning are two techniques which can be employed to deal with inconsistency.
- Deceit: This is when the producer of the information is intentionally misleading the consumer of the information. Cryptography techniques are currently utilized to alleviate this threat.

This list of challenges is illustrative rather than exhaustive, and it focuses on the challenges to the "unifying logic" and "proof" layers of the Semantic Web. The World Wide Web Consortium (W3C) Incubator Group for Uncertainty Reasoning for the World Wide Web (URW3-XG) final report lumps these problems together under the single heading of "uncertainty". Many of the techniques mentioned here will require extensions to the Web Ontology Language (OWL) for example to annotate conditional probabilities. This is an area of active research.

Projects

Here we, lists some of the many projects and tools that exist to create Semantic Web solutions.

DBpedia

DBpedia is an effort to publish structured data: the data is published in RDF and made available on the Web for use under the GNU Free Documentation License, thus allowing

Semantic Web agents to provide inferencing and advanced querying over the Wikipedia-derived dataset and facilitating interlinking, re-use and extension in other data-sources.

FOAF

A popular application of the semantic web is Friend of a Friend (or FoaF), which uses RDF to describe the relationships people have to other people and the "things" around them. FOAF permits intelligent agents to make sense of the thousands of connections people have with each other, their jobs and the items important to their lives; connections that may or may not be enumerated in searches using traditional web search engines. Because the connections are so vast in number, human interpretation of the information may not be the best way of analyzing them.

FOAF is an example of how the Semantic Web attempts to make use of the relationships within a social context.

GoodRelations for e-commerce

A huge potential for Semantic Web technologies lies in adding data structure and typed links to the vast amount of offer data, product model features, and tendering / request for quotation data.

The GoodRelations ontology is a popular vocabulary for expressing product information, prices, payment options, etc. It also allows expressing demand in a straightforward fashion.

GoodRelations has been adopted by Google, BestBuy, Overstock, Yahoo, OpenLink Software, O'Reilly Media, the Book Mashup, and many others.

SIOC

The Semantically-Interlinked Online Communities project (SIOC, pronounced "shock") provides a vocabulary of terms and relationships that model web data spaces. Examples of such data spaces include, among others: discussion forums, blogs, blogrolls / feed subscriptions, mailing lists, shared bookmarks and image galleries.

SIMILE

Semantic **I**nteroperability of **M**etadata and **I**nformation in un**L**ike **E**nvironments

SIMILE is a joint project, conducted by the MIT Libraries and MIT CSAIL, which seeks to enhance interoperability among digital assets, schemata/vocabularies/ontologies, meta data, and services.

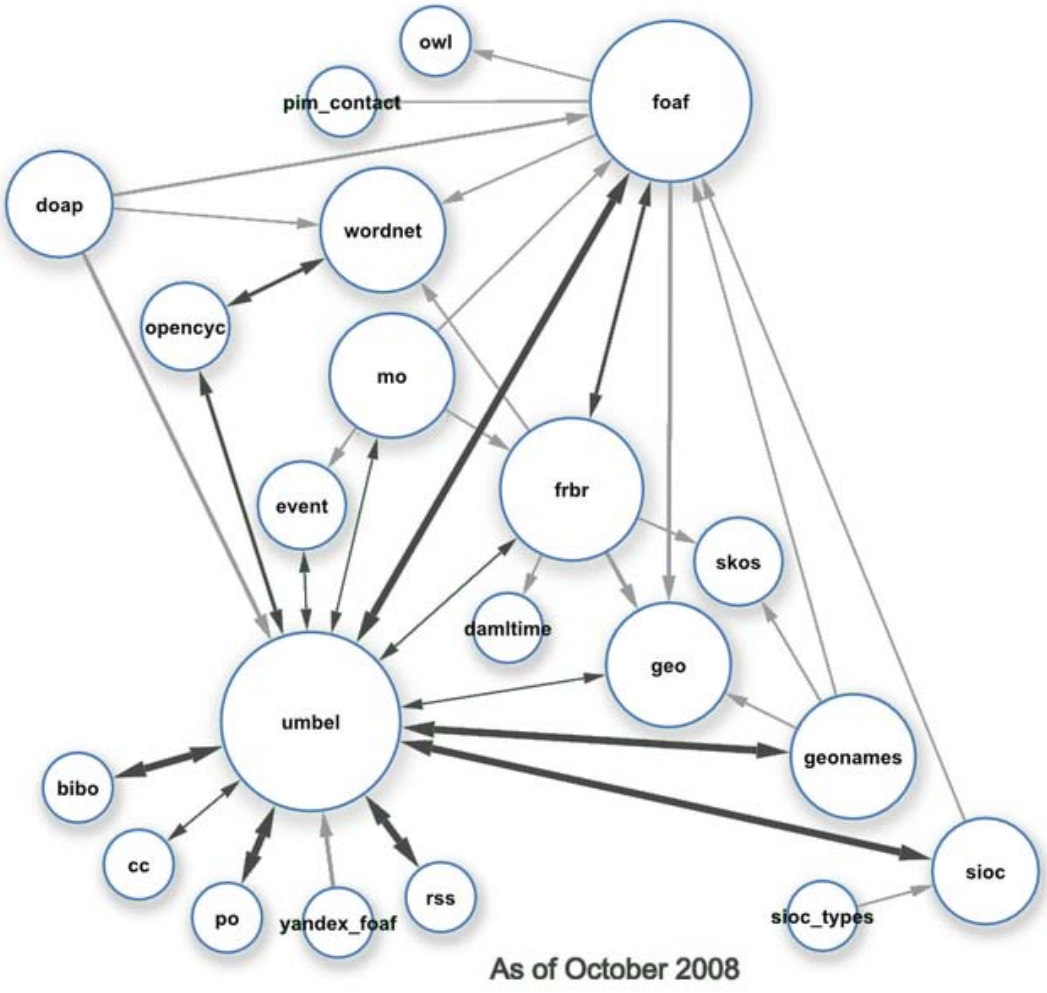
NextBio

A database consolidating high-throughput life sciences experimental data tagged and connected via biomedical ontologies. Nextbio is accessible via a search engine interface.

Researchers can contribute their findings for incorporation to the database. The database currently supports gene or protein expression data and is steadily expanding to support other biological data types.

Linking Open Data

Datasets in the Linking Open Data project, as of Sept 2008



Class linkages within the Linking Open Data datasets

The Linking Open Data project is a W3C-led effort to create openly accessible, and interlinked, RDF Data on the Web. The data in question takes the form of RDF Data Sets drawn from a broad collection of data sources. There is a focus on the Linked Data style of publishing RDF on the Web.

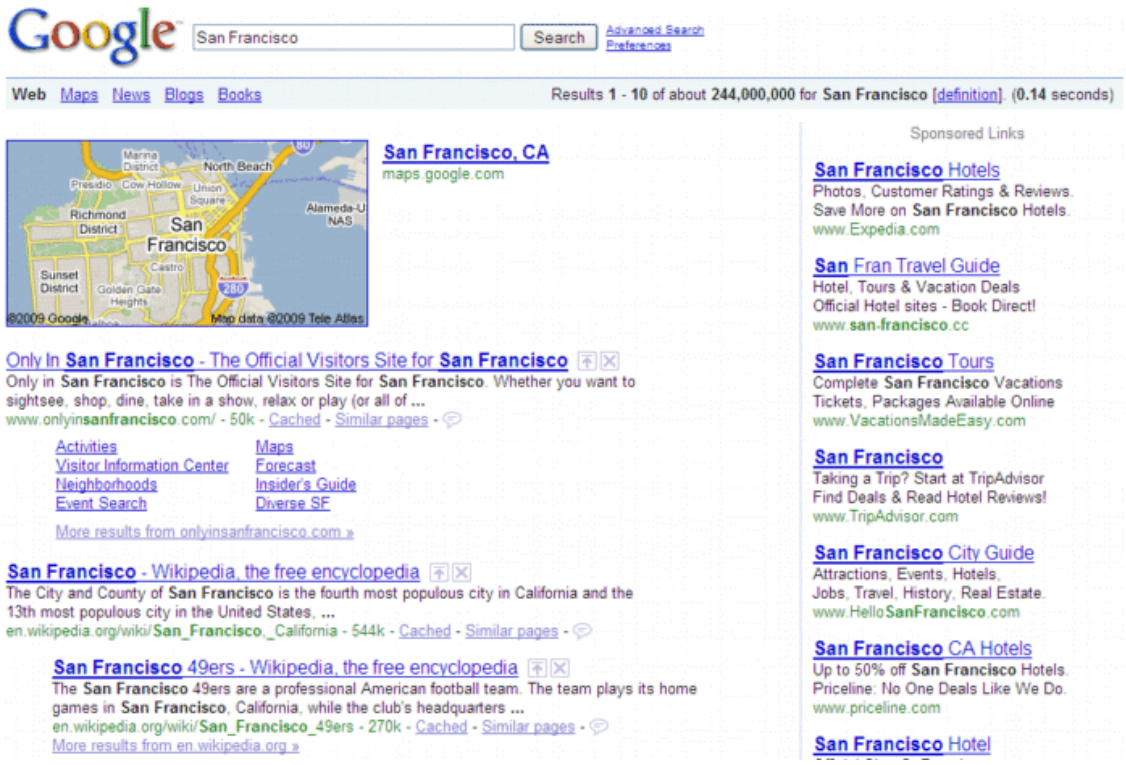
OpenPSI

OpenPSI the (OpenPSI project) is a community effort to create a UK government linked data service that supports research. It is a collaboration between the University of

Southampton and the UK government, led by OPSI at The National Archives and is supported by JISC funding.

Chapter 7

Search Engine Optimization



A typical search engine results page

Search engine optimization (**SEO**) is the process of improving the visibility of a website or a web page in search engines via the "natural" or un-paid ("organic" or "algorithmic") search results. Other forms of search engine marketing (SEM) target paid listings. In general, the earlier (or higher on the page), and more frequently a site appears in the search results list, the more visitors it will receive from the search engine. SEO may target different kinds of search, including image search, local search, video search and industry-specific vertical search engines. This gives a website web presence.

As an Internet marketing strategy, SEO considers how search engines work and what people search for. Optimizing a website may involve editing its content and HTML and associated coding to both increase its relevance to specific keywords and to remove barriers to the indexing activities of search engines. Promoting a site to increase the number of backlinks, or inbound links, is another SEO tactic.

The initialism "SEO" can refer to "search engine optimizers," a term adopted by an industry of consultants who carry out optimization projects on behalf of clients, and by employees who perform SEO services in-house. Search engine optimizers may offer SEO as a stand-alone service or as a part of a broader marketing campaign. Because effective SEO may require changes to the HTML source code of a site and site content, SEO tactics may be incorporated into website development and design. The term "search engine friendly" may be used to describe website designs, menus, content management systems, images, videos, shopping carts, and other elements that have been optimized for the purpose of search engine exposure.

Another class of techniques, known as black hat SEO or spamdexing, uses methods such as link farms, keyword stuffing and article spinning that degrade both the relevance of search results and the user-experience of search engines. Search engines look for sites that employ these techniques in order to remove them from their indices.

History

Webmasters and content providers began optimizing sites for search engines in the mid-1990s, as the first search engines were cataloging the early Web. Initially, all webmasters needed to do was submit the address of a page, or URL, to the various engines which would send a "spider" to "crawl" that page, extract links to other pages from it, and return information found on the page to be indexed. The process involves a search engine spider downloading a page and storing it on the search engine's own server, where a second program, known as an indexer, extracts various information about the page, such as the words it contains and where these are located, as well as any weight for specific words, and all links the page contains, which are then placed into a scheduler for crawling at a later date.

Site owners started to recognize the value of having their sites highly ranked and visible in search engine results, creating an opportunity for both white hat and black hat SEO practitioners. According to industry analyst Danny Sullivan, the phrase "search engine optimization" probably came into use in 1997. The first documented use of the term Search Engine Optimization was John Audette and his company Multimedia Marketing Group as documented by a web page from the MMG site from August, 1997 on the Internet Way Back machine (Document Number 19970801004204). The first registered USA Copyright of a website containing that phrase is by Bruce Clay effective March, 1997 (Document Registration Number TX0005001745, US Library of Congress Copyright Office).

Early versions of search algorithms relied on webmaster-provided information such as the keyword meta tag, or index files in engines like ALIWEB. Meta tags provide a guide to each page's content. Using meta data to index pages was found to be less than reliable,

however, because the webmaster's choice of keywords in the meta tag could potentially be an inaccurate representation of the site's actual content. Inaccurate, incomplete, and inconsistent data in meta tags could and did cause pages to rank for irrelevant searches. Web content providers also manipulated a number of attributes within the HTML source of a page in an attempt to rank well in search engines.

By relying so much on factors such as keyword density which were exclusively within a webmaster's control, early search engines suffered from abuse and ranking manipulation. To provide better results to their users, search engines had to adapt to ensure their results pages showed the most relevant search results, rather than unrelated pages stuffed with numerous keywords by unscrupulous webmasters. Since the success and popularity of a search engine is determined by its ability to produce the most relevant results to any given search, allowing those results to be false would turn users to find other search sources. Search engines responded by developing more complex ranking algorithms, taking into account additional factors that were more difficult for webmasters to manipulate.

Graduate students at Stanford University, Larry Page and Sergey Brin, developed "backrub," a search engine that relied on a mathematical algorithm to rate the prominence of web pages. The number calculated by the algorithm, PageRank, is a function of the quantity and strength of inbound links. PageRank estimates the likelihood that a given page will be reached by a web user who randomly surfs the web, and follows links from one page to another. In effect, this means that some links are stronger than others, as a higher PageRank page is more likely to be reached by the random surfer.

Page and Brin founded Google in 1998. Google attracted a loyal following among the growing number of Internet users, who liked its simple design. Off-page factors (such as PageRank and hyperlink analysis) were considered as well as on-page factors (such as keyword frequency, meta tags, headings, links and site structure) to enable Google to avoid the kind of manipulation seen in search engines that only considered on-page factors for their rankings. Although PageRank was more difficult to game, webmasters had already developed link building tools and schemes to influence the Inktomi search engine, and these methods proved similarly applicable to gaming PageRank. Many sites focused on exchanging, buying, and selling links, often on a massive scale. Some of these schemes, or link farms, involved the creation of thousands of sites for the sole purpose of link spamming.

By 2004, search engines had incorporated a wide range of undisclosed factors in their ranking algorithms to reduce the impact of link manipulation. Google says it ranks sites using more than 200 different signals. The leading search engines, Google and Yahoo, do not disclose the algorithms they use to rank pages. Notable SEO service providers, such as Rand Fishkin, Barry Schwartz, Aaron Wall and Jill Whalen, have studied different approaches to search engine optimization, and have published their opinions in online forums and blogs. SEO practitioners may also study patents held by various search engines to gain insight into the algorithms.

In 2005 Google began personalizing search results for each user. Depending on their history of previous searches, Google crafted results for logged in users. In 2008, Bruce

Clay said that "ranking is dead" because of personalized search. It would become meaningless to discuss how a website ranked, because its rank would potentially be different for each user and each search.

In 2007 Google announced a campaign against paid links that transfer PageRank. On June 15, 2009, Google disclosed that they had taken measures to mitigate the effects of PageRank sculpting by use of the nofollow attribute on links. Matt Cutts, a well-known software engineer at Google, announced that Google Bot would no longer treat nofollowed links in the same way, in order to prevent SEO service providers from using nofollow for PageRank sculpting. As a result of this change the usage of nofollow leads to evaporation of pagerank. In order to avoid the above, SEO engineers developed alternative techniques that replace nofollowed tags with obfuscated Javascript and thus permit PageRank sculpting. Additionally several solutions have been suggested that include the usage of iframes, Flash and Javascript.

In December 2009 Google announced it would be using the web search history of all its users in order to populate search results.

Real-time-search was introduced in late 2009 in an attempt to make search results more timely and relevant. Historically site administrators have spent months or even years optimizing a website to increase search rankings. With the growth in popularity of social media sites and blogs the leading engines made changes to their algorithms to allow fresh content to rank quickly within the search results.

Relationship with search engines

By 1997 search engines recognized that webmasters were making efforts to rank well in their search engines, and that some webmasters were even manipulating their rankings in search results by stuffing pages with excessive or irrelevant keywords. Early search engines, such as Infoseek, adjusted their algorithms in an effort to prevent webmasters from manipulating rankings.

Due to the high marketing value of targeted search results, there is potential for an adversarial relationship between search engines and SEO service providers. In 2005, an annual conference, AIRWeb, Adversarial Information Retrieval on the Web, was created to discuss and minimize the damaging effects of aggressive web content providers.

Companies that employ overly aggressive techniques can get their client websites banned from the search results. In 2005, the Wall Street Journal reported on a company, Traffic Power, which allegedly used high-risk techniques and failed to disclose those risks to its clients. Wired magazine reported that the same company sued blogger and SEO Aaron Wall for writing about the ban. Google's Matt Cutts later confirmed that Google did in fact ban Traffic Power and some of its clients.

Some search engines have also reached out to the SEO industry, and are frequent sponsors and guests at SEO conferences, chats, and seminars. In fact, with the advent of paid inclusion, some search engines now have a vested interest in the health of the optimization community. Major search engines provide information and guidelines to

help with site optimization. Google has a Sitemaps program to help webmasters learn if Google is having any problems indexing their website and also provides data on Google traffic to the website. Google guidelines are a list of suggested practices Google has provided as guidance to webmasters. Yahoo! Site Explorer provides a way for webmasters to submit URLs, determine how many pages are in the Yahoo! index and view link information. Bing Toolbox provides a way from webmasters to submit a sitemap and web feeds, allowing users to determine the crawl rate, and how many pages have been indexed by their search engine.

Methods

Getting indexed

The leading search engines, such as Google, Bing and Yahoo!, use crawlers to find pages for their algorithmic search results. Pages that are linked from other search engine indexed pages do not need to be submitted because they are found automatically. Some search engines, notably Yahoo!, operate a paid submission service that guarantee crawling for either a set fee or cost per click. Such programs usually guarantee inclusion in the database, but do not guarantee specific ranking within the search results. Two major directories, the Yahoo Directory and the Open Directory Project both require manual submission and human editorial review. Google offers Google Webmaster Tools, for which an XML Sitemap feed can be created and submitted for free to ensure that all pages are found, especially pages that aren't discoverable by automatically following links.

Search engine crawlers may look at a number of different factors when crawling a site. Not every page is indexed by the search engines. Distance of pages from the root directory of a site may also be a factor in whether or not pages get crawled.

Preventing crawling

To avoid undesirable content in the search indexes, webmasters can instruct spiders not to crawl certain files or directories through the standard robots.txt file in the root directory of the domain. Additionally, a page can be explicitly excluded from a search engine's database by using a meta tag specific to robots. When a search engine visits a site, the robots.txt located in the root directory is the first file crawled. The robots.txt file is then parsed, and will instruct the robot as to which pages are not to be crawled. As a search engine crawler may keep a cached copy of this file, it may on occasion crawl pages a webmaster does not wish crawled. Pages typically prevented from being crawled include login specific pages such as shopping carts and user-specific content such as search results from internal searches. In March 2007, Google warned webmasters that they should prevent indexing of internal search results because those pages are considered search spam.

Increasing prominence

A variety of methods can increase the prominence of a webpage within the search results. Cross linking between pages of the same website to provide more links to most important pages may improve its visibility. Writing content that includes frequently searched keyword phrase, so as to be relevant to a wide variety of search queries will tend to increase traffic. Updating content so as to keep search engines crawling back frequently can give additional weight to a site. Adding relevant keywords to a web page's meta data, including the title tag and meta description, will tend to improve the relevancy of a site's search listings, thus increasing traffic. URL normalization of web pages accessible via multiple urls, using the "canonical" meta tag or via 301 redirects can help make sure links to different versions of the url all count towards the page's link popularity score.

White hat versus black hat

SEO techniques are classified by some into two broad categories: techniques that search engines recommend as part of good design, and those techniques that search engines do not approve of and attempt to minimize the effect of, referred to as spamdexing. Some industry commentators classify these methods, and the practitioners who employ them, as either white hat SEO, or black hat SEO. White hats tend to produce results that last a long time, whereas black hats anticipate that their sites will eventually be banned once the search engines discover what they are doing.

A SEO tactic, technique or method is considered white hat if it conforms to the search engines' guidelines and involves no deception. As the search engine guidelines are not written as a series of rules or commandments, this is an important distinction to note. White hat SEO is not just about following guidelines, but is about ensuring that the content a search engine indexes and subsequently ranks is the same content a user will see.

White hat advice is generally summed up as creating content for users, not for search engines, and then making that content easily accessible to the spiders, rather than attempting to game the algorithm. White hat SEO is in many ways similar to web development that promotes accessibility, although the two are not identical.

White Hat SEO is merely effective marketing, making efforts to deliver quality content to an audience that has requested the quality content. Traditional marketing means have allowed this through transparency and exposure. A search engine's algorithm takes this into account, such as Google's PageRank.

Black hat SEO attempts to improve rankings in ways that are disapproved of by the search engines, or involve deception. One black hat technique uses text that is hidden, either as text colored similar to the background, in an invisible div, or positioned off screen. Another method gives a different page depending on whether the page is being requested by a human visitor or a search engine, a technique known as cloaking.

Search engines may penalize sites they discover using black hat methods, either by reducing their rankings or eliminating their listings from their databases altogether. Such

penalties can be applied either automatically by the search engines' algorithms, or by a manual site review. One infamous example was the February 2006 Google removal of both BMW Germany and Ricoh Germany for use of deceptive practices. Both companies, however, quickly apologized, fixed the offending pages, and were restored to Google's list.

As a marketing strategy

SEO is not necessarily an appropriate strategy for every website, and other Internet marketing strategies can be much more effective, depending on the site operator's goals. A successful Internet marketing campaign may drive organic traffic, achieved through optimization techniques and not paid advertising, to web pages, but it also may involve the use of paid advertising on search engines and other pages, building high quality web pages to engage and persuade, addressing technical issues that may keep search engines from crawling and indexing those sites, setting up analytics programs to enable site owners to measure their successes, and improving a site's conversion rate.

SEO may generate a return on investment. However, search engines are not paid for organic search traffic, their algorithms change, and there are no guarantees of continued referrals. (Some trading sites such as eBay can be a special case for this; it will announce how and when the ranking algorithm will change a few months before changing the algorithm). Due to this lack of guarantees and certainty, a business that relies heavily on search engine traffic can suffer major losses if the search engines stop sending visitors. It is considered wise business practice for website operators to liberate themselves from dependence on search engine traffic. A top-ranked SEO blog Seomoz.org has suggested, "Search marketers, in a twist of irony, receive a very small share of their traffic from search engines." Instead, their main sources of traffic are links from other websites.

International markets

Optimization techniques are highly tuned to the dominant search engines in the target market. The search engines' market shares vary from market to market, as does competition. In 2003, Danny Sullivan stated that Google represented about 75% of all searches. In markets outside the United States, Google's share is often larger, and Google remains the dominant search engine worldwide as of 2007. As of 2006, Google had an 85-90% market share in Germany. While there were hundreds of SEO firms in the US at that time, there were only about five in Germany. As of June 2008, the marketshare of Google in the UK was close to 90% according to Hitwise. That market share is achieved in a number of countries.

As of 2009, there are only a few large markets where Google is not the leading search engine. In most cases, when Google is not leading in a given market, it is lagging behind a local player. The most notable markets where this is the case are China, Japan, South Korea, Russia and the Czech Republic where respectively Baidu, Yahoo! Japan, Naver, Yandex and Seznam are market leaders.

Successful search optimization for international markets may require professional translation of web pages, registration of a domain name with a top level domain in the target market, and web hosting that provides a local IP address. Otherwise, the fundamental elements of search optimization are essentially the same, regardless of language.

Legal precedents

On October 17, 2002, SearchKing filed suit in the United States District Court, Western District of Oklahoma, against the search engine Google. SearchKing's claim was that Google's tactics to prevent spamdexing constituted a tortious interference with contractual relations. On May 27, 2003, the court granted Google's motion to dismiss the complaint because SearchKing "failed to state a claim upon which relief may be granted."

In March 2006, KinderStart filed a lawsuit against Google over search engine rankings. Kinderstart's website was removed from Google's index prior to the lawsuit and the amount of traffic to the site dropped by 70%. On March 16, 2007 the United States District Court for the Northern District of California (San Jose Division) dismissed KinderStart's complaint without leave to amend, and partially granted Google's motion for Rule 11 sanctions against KinderStart's attorney, requiring him to pay part of Google's legal expenses.

Chapter 8

Hypertext Transfer Protocol

The **Hypertext Transfer Protocol** (**HTTP**) is a networking protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

The standards development of HTTP has been coordinated by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs), most notably RFC 2616 (June 1999), which defines HTTP/1.1, the version of HTTP in common use.

Technical overview

HTTP functions as a request-response protocol in the client-server computing model. In HTTP, a web browser, for example, acts as a *client*, while an application running on a computer hosting a web site functions as a *server*. The client submits an HTTP *request* message to the server. The server, which stores content, or provides *resources*, such as HTML files, or performs other functions on behalf of the client, returns a response message to the client. A response contains completion status information about the request and may contain any content requested by the client in its message body.

A client is often referred to as a *user agent* (UA). A web browser, or web crawler (*spider, used by search providers*) are examples of common types of clients or user agents.

The HTTP protocol is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of the original, so-called *origin server* to improve response time. HTTP proxy servers at network boundaries facilitate communication when clients without a globally routable address are located in private networks by relaying the requests and responses between clients and servers.

HTTP is an Application Layer protocol designed within the framework of the Internet Protocol Suite. The protocol definitions presume a reliable Transport Layer protocol for host-to-host data transfer. The Transmission Control Protocol (TCP) is the dominant

protocol in use for this purpose. However, HTTP has found application even with unreliable protocols, such as the User Datagram Protocol (UDP) in methods such as the Simple Service Discovery Protocol (SSDP).

HTTP Resources are identified and located on the network by Uniform Resource Identifiers (URIs)—or, more specifically, Uniform Resource Locators (URLs)—using the http or https URI schemes. URIs and the Hypertext Markup Language (HTML), form a system of inter-linked resources, called hypertext documents, on the Internet, that led to the establishment of the World Wide Web in 1990 by English physicist Tim Berners-Lee.

The original version of HTTP (HTTP/1.0) was revised in HTTP/1.1. HTTP/1.0 uses a separate connection to the same server for every request-response transaction, while HTTP/1.1 can reuse a connection multiple times, to download, for instance, images for a just delivered page. Hence HTTP/1.1 communications experience less latency as the establishment of TCP connections presents considerable overhead.

History

The term HyperText was coined by Ted Nelson who in turn was inspired by Vannevar Bush's microfilm-based "memex". Tim Berners-Lee first proposed the "WorldWideWeb" project — now known as the World Wide Web. Berners-Lee and his team are credited with inventing the original HTTP protocol along with the HTML and the associated technology for a web server and a text-based web browser. The first version of the protocol had only one method, namely GET, which would request a page from a server. The response from the server was always an HTML page.

The first documented version of HTTP was HTTP V0.9 (1991). Dave Raggett led the HTTP Working Group (HTTP WG) in 1995 and wanted to expand the protocol extended operations, extended negotiation, richer meta-information, tied with a security protocol and got more efficient by adding additional methods and header fields. RFC 1945 officially introduced and recognized HTTP V1.0 in 1996.

The HTTP WG planned to publish new standards in December 1995 and the support for pre-standard HTTP/1.1 based on the then developing RFC 2068 (called HTTP-NG) was rapidly adopted by the major browser developers in early 1996. By March 1996, pre-standard HTTP/1.1 was supported in Arena, Netscape 2.0, Netscape Navigator Gold 2.01, Mosaic 2.7, Lynx 2.5, and in Internet Explorer 3.0. End user adoption of the new browsers was rapid. In March 1996, one web hosting company reported that over 40% of browsers in use on the Internet were HTTP 1.1 compliant. That same web hosting company reported that by June 1996, 65% of all browsers accessing their servers were HTTP/1.1 compliant. The HTTP/1.1 standard as defined in RFC 2068 was officially released in January 1997. Improvements and updates to the HTTP/1.1 standard were released under RFC 2616 in June 1999.

HTTP session

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request. It establishes a Transmission Control Protocol (TCP) connection to a particular port on a host. An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own, the body of which is perhaps the requested resource, an error message, or some other information.

Request message

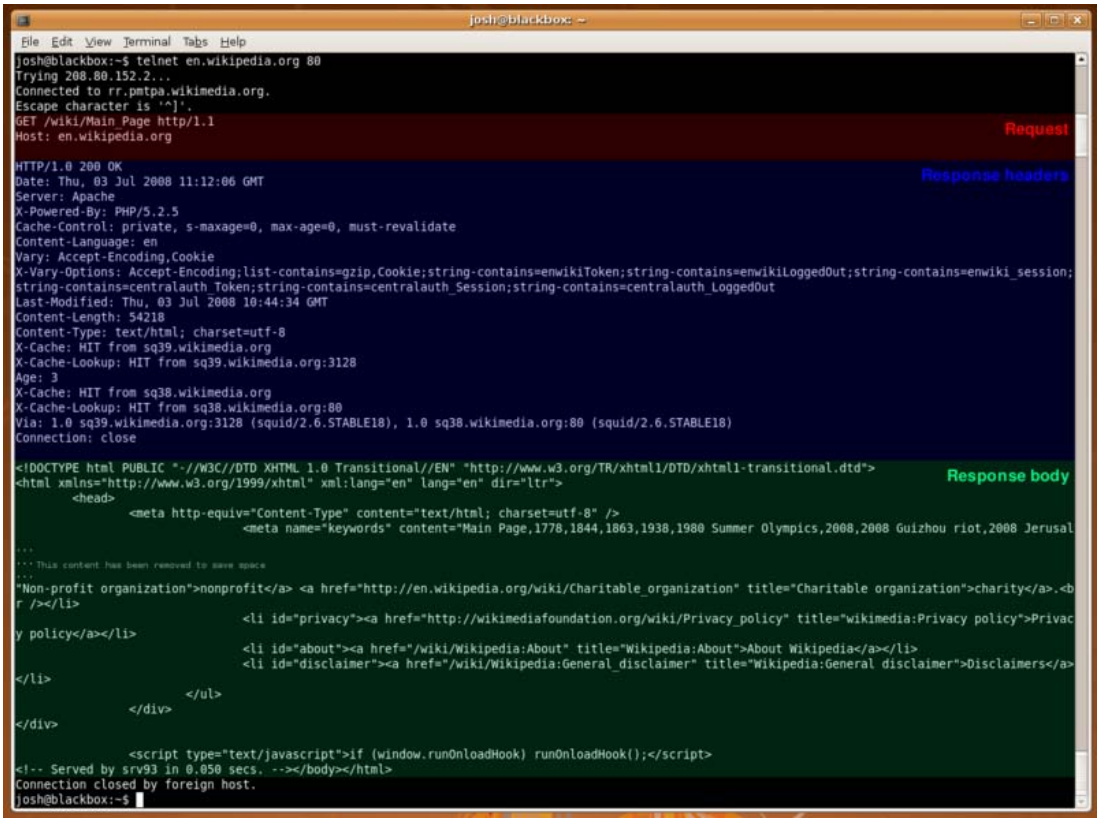
The request message consists of the following:

- Request line, such as GET /images/logo.png HTTP/1.1, which requests a resource called /images/logo.png from server
- Headers, such as Accept-Language: en
- An empty line
- An optional message body

The request line and headers must all end with <CR><LF> (that is, a carriage return followed by a line feed). The empty line must consist of only <CR><LF> and no other whitespace. In the HTTP/1.1 protocol, all headers except Host are optional.

A request line containing only the path name is accepted by servers to maintain compatibility with HTTP clients before the HTTP/1.0 specification in RFC1945.

Request methods



An HTTP request made using telnet. The request, response headers and response body are highlighted.

HTTP defines nine methods (sometimes referred to as "verbs") indicating the desired action to be performed on the identified **resource**. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

HEAD
: Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

GET
: Requests a representation of the specified resource. Requests using GET (and a few other HTTP methods) "SHOULD NOT have the significance of taking an action other than retrieval". The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations.".

POST

Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.

PUT
: Uploads a representation of the specified resource.

DELETE
: Deletes the specified resource.

TRACE
: Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.

OPTIONS
: Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

CONNECT
: Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

PATCH
: Is used to apply partial modifications to a resource.

HTTP servers are required to implement at least the GET and HEAD methods and, whenever possible, also the OPTIONS method.

Safe methods

Some methods (for example, HEAD, GET, OPTIONS and TRACE) are defined as *safe*, which means they are intended only for information retrieval and should not change the state of the server. In other words, they should not have side effects, beyond relatively harmless effects such as logging, caching, the serving of banner advertisements or incrementing a web counter. Making arbitrary GET requests without regard to the context of the application's state should therefore be considered safe.

By contrast, methods such as POST, PUT and DELETE are intended for actions that may cause side effects either on the server, or external side effects such as financial transactions or transmission of email. Such methods are therefore not usually used by conforming web robots or web crawlers, which tend to make requests without regard to context or consequences.

Despite the prescribed safety of *GET* requests, in practice their handling by the server is not technically limited in any way. Therefore, careless or deliberate programming can cause non-trivial changes on the server. This is discouraged, because it can cause problems for Web caching, search engines and other automated agents, which can make unintended changes on the server.

Furthermore, methods such as TRACE, TRACK and DEBUG are considered potentially 'unsafe' by some security professionals, because they can be used by attackers to gather information or bypass security controls during attacks. Security software tools such as

Tenable Nessus and Microsoft URLScan report on the presence of these methods as being security issues.

Idempotent methods and web applications

Methods PUT and DELETE are defined to be idempotent, meaning that multiple identical requests should have the same effect as a single request. Methods GET, HEAD, OPTIONS and TRACE, being prescribed as safe, should also be idempotent, as HTTP is a stateless protocol.

In contrast, the POST method is not necessarily idempotent, and therefore sending an identical POST request multiple times may further affect state or cause further side effects (such as financial transactions). In some cases this may be desirable, but in other cases this could be due to an accident, such as when a user does not realize that their action will result in sending another request, or they did not receive adequate feedback that their first request was successful. While web browsers may show alert dialog boxes to warn users in some cases where reloading a page may re-submit a POST request, it is generally up to the web application to handle cases where a POST request should not be submitted more than once.

Note that whether a method is idempotent is not enforced by the protocol or web server. It is perfectly possible to write a web application in which (for example) a database insert or other non-idempotent action is triggered by a GET or other request. Ignoring this recommendation, however, may result in undesirable consequences, if a user agent assumes that repeating the same request is safe when it isn't.

Status codes

In HTTP/1.0 and since, the first line of the HTTP response is called the *status line* and includes a numeric *status code* (such as "404") and a textual *reason phrase* (such as "Not Found"). The way the user agent handles the response primarily depends on the code and secondarily on the response headers. Custom status codes can be used since, if the user agent encounters a code it does not recognize, it can use the first digit of the code to determine the general class of the response.

Also, the standard *reason phrases* are only recommendations and can be replaced with "local equivalents" at the web developer's discretion. If the status code indicated a problem, the user agent might display the *reason phrase* to the user to provide further information about the nature of the problem. The standard also allows the user agent to attempt to interpret the *reason phrase*, though this might be unwise since the standard explicitly specifies that status codes are machine-readable and *reason phrases* are human-readable.

Persistent connections

In HTTP/0.9 and 1.0, the connection is closed after a single request/response pair. In HTTP/1.1 a keep-alive-mechanism was introduced, where a connection could be reused for more than one request.

Such *persistent connections* reduce lag perceptibly, because the client does not need to re-negotiate the TCP connection after the first request has been sent.

Version 1.1 of the protocol made bandwidth optimization improvements to HTTP/1.0. For example, HTTP/1.1 introduced chunked transfer encoding to allow content on persistent connections to be streamed, rather than buffered. HTTP pipelining further reduces lag time, allowing clients to send multiple requests before a previous response has been received to the first one. Another improvement to the protocol was byte serving, which is when a server transmits just the portion of a resource explicitly requested by a client.

HTTP session state

HTTP is a stateless protocol. A stateless protocol does not require the server to retain information or status about each user for the duration of multiple requests. For example, when a web server is required to customize the content of a web page for a user, the web application may have to track the user's progress from page to page. A common solution is the use of HTTP cookies. Other methods include server side sessions, hidden variables (when the current page is a form), and URL-rewriting using URI-encoded parameters, e.g., /index.php?session_id=some_unique_session_code.

Secure HTTP

There are currently two methods of establishing a secure HTTP connection: the https URI scheme and the HTTP 1.1 Upgrade header, introduced by RFC 2817. Browser support for the Upgrade header is, however, nearly non-existent, so HTTPS is still the dominant method of establishing a secure HTTP connection. Secure HTTP is notated by the prefix https:// instead of http:// on web URIs.

https URI scheme

https is a URI scheme that is, aside from the scheme token, syntactically identical to the http scheme used for normal HTTP connections, but which signals the browser to use an added encryption layer of SSL/TLS to protect the traffic. SSL is especially suited for HTTP since it can provide some protection even if only one side of the communication is authenticated. This is the case with HTTP transactions over the Internet, where typically only the server is authenticated (by the client examining the server's certificate).

HTTP 1.1 Upgrade header field

HTTP 1.1 introduced support for the Upgrade header field. In the exchange, the client begins by making a clear-text request, which is later upgraded to Transport Layer Security (TLS). Either the client or the server may request that the connection be upgraded. The most common usage is a clear-text request by the client followed by a server demand to upgrade the connection:

Client:

```
GET /encrypted-area HTTP/1.1
Host: www.example.com
```

Server:

```
HTTP/1.1 426 Upgrade Required
Upgrade: TLS/1.0, HTTP/1.1
Connection: Upgrade
```

The server returns a 426 status-code to alert legacy clients that the failure was client-related (400 level codes indicate a client failure: List of HTTP status codes).

This method for establishing a secure connection is advantageous because it:

- Does not require messy and problematic redirection and URL rewriting on the server side.
- Enables virtual hosting of secured websites (although HTTPS also allows this using Server Name Indication).
- Reduces the potential for user confusion by providing a single way to access a particular resource.

A disadvantage of this method is that the client cannot specify the requirement for a secure HTTP in the URI. Thus, the (untrusted) server will be responsible for enabling secure HTTP, not the (trusted) client.

Example session

Below is a sample conversation between an HTTP client and an HTTP server running on www.example.com, port 80.

Client request

```
GET /index.html HTTP/1.1
Host: www.example.com
```

A client request (consisting in this case of the request line and only one header) is followed by a blank line, so that the request ends with a double newline, each in the form

of a carriage return followed by a line feed. The "Host" header distinguishes between various DNS names sharing a single IP address, allowing name-based virtual hosting. While optional in HTTP/1.0, it is mandatory in HTTP/1.1.

Server response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

A server response is followed by a blank line and text of the requested page. The ETag (entity tag) header is used to determine if a cached version of the requested resource is identical to the current version of the resource on the server. *Content-Type* specifies the Internet media type of the data conveyed by the http message, while *Content-Length* indicates its length in bytes. The HTTP/1.1 webserver publishes its ability to respond to requests for certain byte ranges of the document by setting the header *Accept-Ranges: bytes*. This is useful, if the client needs to have only certain portions of a resource sent by the server, which is called byte serving. When *Connection: close* is sent in a header, it means that the web server will close the TCP connection immediately after the transfer of this response.

Chapter 9

Web Server

The inside and front of a Dell PowerEdge Web server

A **web server** can be referred to as either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed through the Internet.

The most common use of Web servers is to host Web sites but there are other uses like data storage or for running enterprise applications.

Overview

The primary function of a web server is to deliver web pages on the request to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, style sheets and JavaScripts.

A client, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. The resource is typically a real file on the server's secondary memory, but this is not necessarily the case and depends on how the web server is implemented.

While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.

Many generic web servers also support server-side scripting, e.g., Apache HTTP Server and PHP. This means that the behaviour of the web server can be scripted in separate files, while the actual server software remains unchanged. Usually, this function is used to create HTML documents "on-the-fly" as opposed to returning fixed documents. This is referred to as dynamic and static content respectively. The former is primarily used for retrieving and/or modifying information from databases. The latter is, however, typically much faster and more easily cached.

Web servers are not always used for serving the world wide web. They can also be found embedded in devices such as printers, routers, webcams and serving only a local network. The web server may then be used as a part of a system for monitoring and/or administrating the device in question. This usually means that no additional software has to be installed on the client computer, since only a web browser is required (which now is included with most operating systems).

History of web servers

The world's first web server.

In 1989 Tim Berners-Lee proposed to his employer CERN (European Organization for Nuclear Research) a new project, which had the goal of easing the exchange of information between scientists by using a hypertext system. As a result of the implementation of this project, in 1990 Berners-Lee wrote two programs:

- a browser called WorldWideWeb;
- the world's first web server, later known as CERN httpd, which ran on NeXTSTEP.

Between 1991 and 1994 the simplicity and effectiveness of early technologies used to surf and exchange data through the World Wide Web helped to port them to many different operating systems and spread their use among lots of different social groups of people, first in scientific organizations, then in universities and finally in industry.

In 1994 Tim Berners-Lee decided to constitute the World Wide Web Consortium (W3C) to regulate the further development of the many technologies involved (HTTP, HTML, etc.) through a standardization process.

Common features

1. **Virtual hosting** to serve many Web sites using one IP address.
2. **Large file support** to be able to serve files whose size is greater than 2 GB on 32 bit OS.
3. **Bandwidth throttling** to limit the speed of responses in order to not saturate the network and to be able to serve more clients.
4. **Server-side scripting** to generate dynamic Web pages, still keeping Web server and Web site implementations separate from each other.

Path translation

Web servers are able to map the path component of a Uniform Resource Locator (**URL**) into:

- a local file system resource (for static requests);
- an internal or external program name (for dynamic requests).

For a *static request* the URL path specified by the client is relative to the Web server's root directory.

Consider the following URL as it would be requested by a client:

```
http://www.example.com/path/file.html
```

The client's user agent will translate it into a connection to `www.example.com` with the following HTTP 1.1 request:

```
GET /path/file.html HTTP/1.1
Host: www.example.com
```

The Web server on `www.example.com` will append the given path to the path of its root directory. On an Apache server, this is commonly `/home/www` (On Unix machines, usually `/var/www`). The result is the local file system resource:

```
/home/www/path/file.html
```

The Web server then reads the file, if it exists and sends a response to the client's Web browser. The response will describe the content of the file and contain the file itself or an error message will return saying that the file does not exist or is unavailable.

Load limits

A Web server (program) has defined load limits, because it can handle only a limited number of concurrent client connections (usually between 2 and 80,000, by default between 500 and 1,000) per IP address (and TCP port) and it can serve only a certain maximum number of requests per second depending on:

- its own settings;
- the HTTP request type;
- content origin (static or dynamic);
- the fact that the served content is or is not cached;
- the hardware and software limitations of the OS where it is working;

When a Web server is near to or over its limits, it becomes unresponsive.

Kernel-mode and user-mode Web servers

A Web server can be either implemented into the OS kernel, or in user space (like other regular applications).

An in-kernel Web server (like TUX on GNU/Linux or Microsoft IIS on Windows) will usually work faster, because, as part of the system, it can directly use all the hardware resources it needs, such as non-paged memory, CPU time-slices, network adapters, or buffers.

Web servers that run in user-mode have to ask the system the permission to use more memory or more CPU resources. Not only do these requests to the kernel take time, but they are not always satisfied because the system reserves resources for its own usage and has the responsibility to share hardware resources with all the other running applications.

Also, applications cannot access the system's internal buffers, which causes useless buffer copies that create another handicap for user-mode web servers. As a consequence, the only way for a user-mode web server to match kernel-mode performance is to raise the quality of its code to much higher standards, similar to that of the code used in web servers that run in the kernel. This is a significant issue under Windows, where the user-mode overhead is about six times greater than that under Linux.

Overload causes

At any time web servers can be overloaded because of:

- **Too much legitimate web traffic.** Thousands or even millions of clients connecting to the web site in a short interval, e.g., Slashdot effect;
- **Distributed Denial of Service** attacks;
- **Computer worms** that sometimes cause abnormal traffic because of millions of infected computers (not coordinated among them);
- **XSS viruses** can cause high traffic because of millions of infected browsers and/or Web servers;
- **Internet bots.** Traffic not filtered/limited on large web sites with very few resources (bandwidth, etc.);
- **Internet (network) slowdowns**, so that client requests are served more slowly and the number of connections increases so much that server limits are reached;
- **Web servers (computers) partial unavailability.** This can happen because of required or urgent maintenance or upgrade, hardware or software failures, back-

end (e.g., database) failures, etc.; in these cases the remaining web servers get too much traffic and become overloaded.

Overload symptoms

The symptoms of an overloaded Web server are:

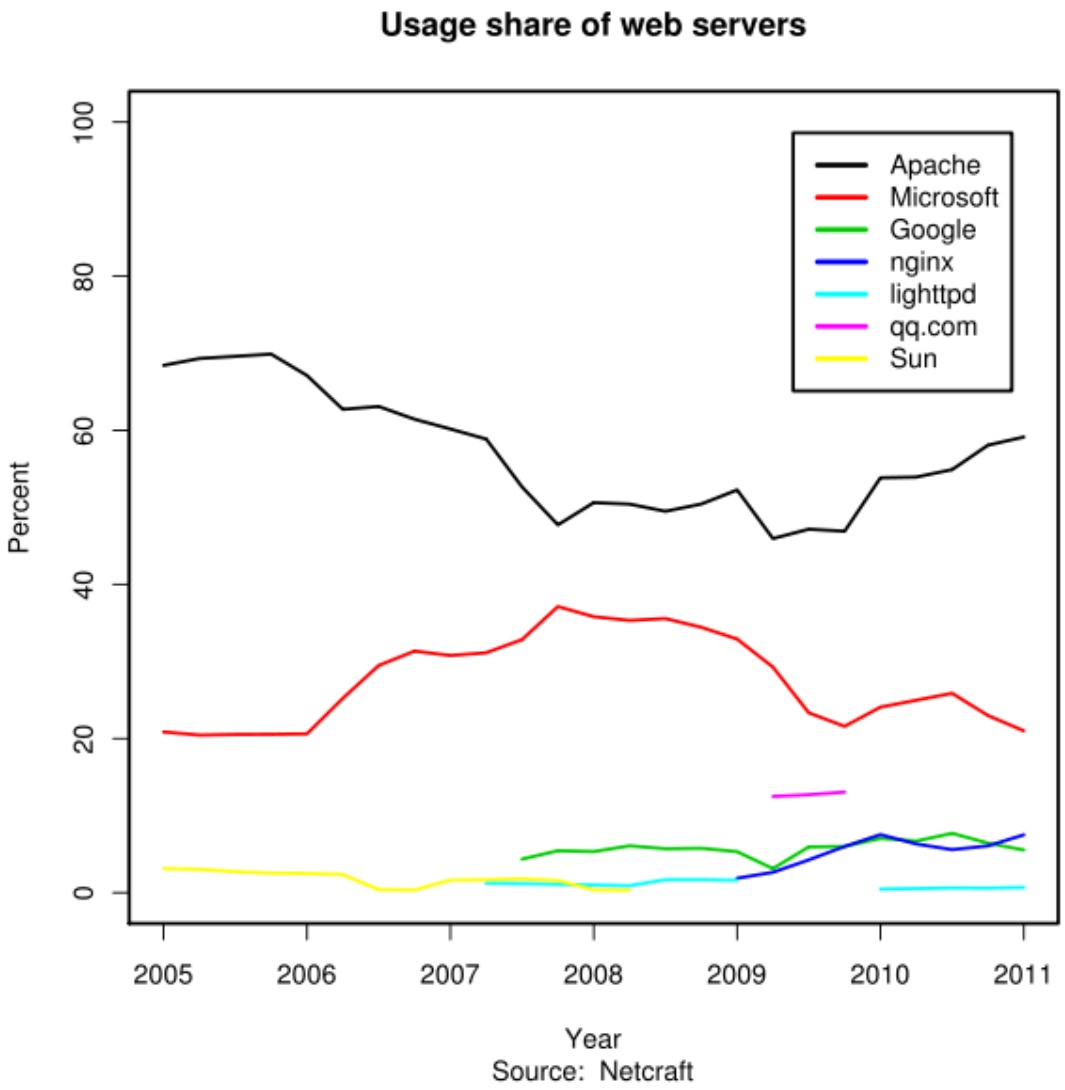
- requests are served with (possibly long) delays (from 1 second to a few hundred seconds);
- 500, 502, 503, 504 HTTP errors are returned to clients (sometimes also unrelated 404 error or even 408 error may be returned);
- TCP connections are refused or reset (interrupted) before any content is sent to clients;
- in very rare cases, only partial contents are sent (but this behavior may well be considered a bug, even if it usually depends on unavailable system resources).

Anti-overload techniques

To partially overcome above load limits and to prevent overload, most popular Web sites use common techniques like:

- **managing network traffic**, by using:
  - **Firewalls** to block unwanted traffic coming from bad IP sources or having bad patterns;
  - **HTTP traffic managers** to drop, redirect or rewrite requests having bad HTTP patterns;
  - **Bandwidth management** and **traffic shaping**, in order to smooth down peaks in network usage;
- deploying **Web cache** techniques;
- using different domain names to serve different (static and dynamic) content by separate Web servers, i.e.:
  - http://images.example.com
  - http://www.example.com
- using different domain names and/or computers to separate big files from small and medium sized files; the idea is to be able to fully cache small and medium sized files and to efficiently serve big or huge (over 10 - 1000 MB) files by using different settings;
- using many Web servers (programs) per computer, each one bound to its own network card and IP address;
- using many Web servers (computers) that are grouped together so that they act or are seen as one big Web server;
- adding more hardware resources (i.e. RAM, disks) to each computer;
- tuning OS parameters for hardware capabilities and usage;
- using more efficient computer programs for Web servers, etc.;
- using other workarounds, especially if dynamic content is involved.

Market structure



Market share of major Web servers

Below is the most recent statistics of the market share of the top web servers on the internet by Netcraft survey in November 2010

Vendor	Product	Web Sites Hosted	Percent
Apache	Apache	148,085,963	59.36%
Microsoft	IIS	56,637,980	22.70%
Igor Sysoev	nginx	15,058,114	6.04%
Google	GWS	14,827,157	5.94%
lighttpd	lighttpd	2,070,300	0.83%

Chapter 10

HTML

HTML (HyperText Markup Language)	
Filename extension	`.html, .htm`
Internet media type	`text/html`
Type code	TEXT
Uniform Type Identifier	public.html
Developed by	World Wide Web Consortium & WHATWG
Type of format	Markup language
Extended from	SGML
Extended to	XHTML
Standard(s)	ISO/IEC 15445 W3C HTML 4.01 W3C HTML5 (draft)

HTML, which stands for **HyperText Markup Language**, is the predominant markup language for web pages. HTML is the basic building-blocks of webpages.

HTML is written in the form of HTML elements consisting of *tags*, enclosed in angle brackets (like <html>), within the web page content. HTML tags normally come in pairs like <h1> and </h1>. The first tag in a pair is the *start tag*, the second tag is the *end tag* (they are also called *opening tags* and *closing tags*).

The purpose of a web browser is to read HTML documents and compose them into visual or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts in languages such as JavaScript which affect the behavior of HTML webpages.

Web browsers can also refer to Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicitly presentational HTML markup.

History

The historic logo made by the W3C.

Origins

Tim Berners-Lee

In 1980, physicist Tim Berners-Lee, who was a contractor at CERN, proposed and prototyped ENQUIRE, a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system. Berners-Lee specified HTML and wrote the browser and server software in the last part of 1990. In that year, Berners-Lee and CERN data systems engineer Robert Cailliau collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes from 1990 he lists "*some of the many areas in which hypertext is used*" and puts an encyclopedia first.

First specifications

The first publicly available description of HTML was a document called *HTML Tags*, first mentioned on the Internet by Berners-Lee in late 1991. It describes 20 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by SGMLguid, an in-house SGML based documentation format at CERN. Thirteen of these elements still exist in HTML 4.

HTML is a markup language that web browsers use to interpret and compose text, images and other material into visual or audible web pages. Default characteristics for every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of CSS. Many of the text elements are found in the 1988 ISO technical report TR 9537 *Techniques for using SGML*, which in turn covers the features of early text formatting languages such as that used by the RUNOFF command developed in the early 1960s for the CTSS (Compatible Time-Sharing System) operating system: these formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with also the separation of structure and processing; HTML has been progressively moved in this direction with CSS.

Berners-Lee considered HTML to be an application of SGML. It was formally defined as such by the Internet Engineering Task Force (IETF) with the mid-1993 publication of the first proposal for an HTML specification: "Hypertext Markup Language (HTML)" Internet-Draft by Berners-Lee and Dan Connolly, which included an SGML Document Type Definition to define the grammar. The draft expired after six months, but was notable for its acknowledgment of the NCSA Mosaic browser's custom tag for embedding in-line images, reflecting the IETF's philosophy of basing standards on successful prototypes. Similarly, Dave Raggett's competing Internet-Draft, "HTML+ (Hypertext Markup Format)", from late 1993, suggested standardizing already-implemented features like tables and fill-out forms.

After the HTML and HTML+ drafts expired in early 1994, the IETF created an HTML Working Group, which in 1995 completed "HTML 2.0", the first HTML specification intended to be treated as a standard against which future implementations should be based. Published as Request for Comments 1866, HTML 2.0 included ideas from the HTML and HTML+ drafts. The 2.0 designation was intended to distinguish the new edition from previous drafts.

Further development under the auspices of the IETF was stalled by competing interests. Since 1996, the HTML specifications have been maintained, with input from commercial software vendors, by the World Wide Web Consortium (W3C). However, in 2000, HTML also became an international standard (ISO/IEC 15445:2000). The last HTML specification published by the W3C is the HTML 4.01 Recommendation, published in late 1999. Its issues and errors were last acknowledged by errata published in 2001.

Version history of the standard

HTML version timeline

November 24, 1995
: HTML 2.0 was published as IETF RFC 1866. Supplemental RFCs added capabilities:

- November 25, 1995: RFC 1867 (form-based file upload)
- May 1996: RFC 1942 (tables)
- August 1996: RFC 1980 (client-side image maps)
- January 1997: RFC 2070 (internationalization)

: In June 2000, all of these were declared obsolete/historic by RFC 2854.

January 1997
: HTML 3.2 was published as a W3C Recommendation. It was the first version developed and standardized exclusively by the W3C, as the IETF had closed its HTML Working Group in September 1996.
: HTML 3.2 dropped math formulas entirely, reconciled overlap among various proprietary extensions and adopted most of Netscape's visual markup tags. Netscape's blink element and Microsoft's marquee element were omitted due to a mutual agreement between the two companies. A markup for mathematical formulas similar to that in HTML wasn't standardized until 14 months later in MathML.

December 1997
: HTML 4.0 was published as a W3C Recommendation. It offers three variations:

- Strict, in which deprecated elements are forbidden,
- Transitional, in which deprecated elements are allowed,
- Frameset, in which mostly only frame related elements are allowed;

: Initially code-named "Cougar", HTML 4.0 adopted many browser-specific element types and attributes, but at the same time sought to phase out Netscape's visual markup features by marking them as deprecated in favor of style sheets.
: HTML 4 is an SGML application conforming to ISO 8879 - SGML.

April 1998
: HTML 4.0 was reissued with minor edits without incrementing the version number.

December 1999
: HTML 4.01 was published as a W3C Recommendation. It offers the same three variations as HTML 4.0 and its last errata were published May 12, 2001.

May 2000
: ISO/IEC 15445:2000 ("ISO HTML", based on HTML 4.01 Strict) was published as an ISO/IEC international standard. In the ISO this standard falls in the domain of the ISO/IEC JTC1/SC34 (ISO/IEC Joint Technical Committee 1, Subcommittee 34 - Document description and processing languages).
: As of mid-2008, HTML 4.01 and ISO/IEC 15445:2000 are the most recent versions of HTML. Development of the parallel, XML-based language XHTML occupied the W3C's HTML Working Group through the early and mid-2000s.

HTML draft version timeline

October 1991
: *HTML Tags*, an informal CERN document listing twelve HTML tags, was first mentioned in public.

June 1992
: First informal draft of the HTML DTD, with seven subsequent revisions (July 15, August 6, August 18, November 17, November 19, November 20, November 22)

November 1992
: HTML DTD 1.1 (the first with a version number, based on RCS revisions, which start with 1.1 rather than 1.0), an informal draft

June 1993
: Hypertext Markup Language was published by the IETF IIIR Working Group as an Internet-Draft (a rough proposal for a standard). It was replaced by a second version one month later, followed by six further drafts published by IETF itself that finally led to HTML 2.0 in RFC1866

November 1993
: HTML+ was published by the IETF as an Internet-Draft and was a competing proposal to the Hypertext Markup Language draft. It expired in May 1994.

April 1995 (authored March 1995)
: HTML 3.0 was proposed as a standard to the IETF, but the proposal expired five months later without further action. It included many of the capabilities that were in Raggett's HTML+ proposal, such as support for tables, text flow around figures and the display of complex mathematical formulas.
: W3C began development of its own Arena browser as a test bed for HTML 3 and Cascading Style Sheets, but HTML 3.0 did not succeed for several reasons. The draft was considered very large at 150 pages and the pace of browser development, as well as the number of interested parties, had outstripped the resources of the IETF. Browser vendors, including Microsoft and Netscape at the time, chose to implement different subsets of HTML 3's draft features as well as to introduce their own extensions to it. These included extensions to control stylistic aspects of documents, contrary to the "belief [of the academic engineering community] that such things as text color, background texture, font size and font face were definitely outside the scope of a language when their only intent was to specify how a document would be organized." Dave Raggett, who has been a W3C Fellow for many years has commented for example, "To a certain extent, Microsoft built its business on the Web by extending HTML features."

January 2008

HTML5 was published as a Working Draft (link) by the W3C.
Although its syntax closely resembles that of SGML, HTML5 has abandoned any attempt to be an SGML application and has explicitly defined its own "html" serialization, in addition to an alternative XML-based XHTML5 serialization.

XHTML versions

XHTML is a separate language that began as a reformulation of HTML 4.01 using XML 1.0. It continues to be developed:

- XHTML 1.0, published January 26, 2000 as a W3C Recommendation, later revised and republished August 1, 2002. It offers the same three variations as HTML 4.0 and 4.01, reformulated in XML, with minor restrictions.
- XHTML 1.1, published May 31, 2001 as a W3C Recommendation. It is based on XHTML 1.0 Strict, but includes minor changes, can be customized, is reformulated using modules from Modularization of XHTML, which was published April 10, 2001 as a W3C Recommendation.
- XHTML 2.0,. There is no XHTML 2.0 standard. XHTML 2.0 is incompatible with XHTML 1.x and, therefore, would be more accurately characterized as an XHTML-inspired new language than an update to XHTML 1.x.
- XHTML5, which is an update to XHTML 1.x, is being defined alongside HTML5 in the HTML5 draft.

Markup

HTML markup consists of several key components, including *elements* (and their *attributes*), character-based *data types*, *character references* and *entity references*. Another important component is the *document type declaration*, which triggers standards mode rendering.

The Hello world program, a common computer program employed for comparing programming languages, scripting languages and markup languages is made of 9 lines of code although in HTML newlines are optional:

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

(The text between <html> and </html> describes the web page, and The text between <body> and </body> is the visible page content.)

This Document Type Declaration is for HTML5. If the `<!doctype html>` declaration is not included, various browsers will revert to "quirks mode" for rendering.

Elements

HTML documents are composed entirely of **HTML elements** that, in their most general form have three components: a pair of element ***tags***, a "start tag" and "end tag"; some element ***attributes*** within the start tag; and finally, any textual and graphical *content* between the start and end tags. The **HTML element** is everything between and including the tags. Each **tag** is enclosed in angle brackets.

The general form of an **HTML element** is therefore: `<tag attribute1="value1" attribute2="value2">content to be rendered</tag>` The name of the HTML element is also the name of the tag. Note that the end tag's name is preceded by a slash character, "/". If attributes are not assigned, default values are used.

Element examples

Header of the HTML document:<head>...</head>. Usually the title should be included in the head, for example:

```
<head>
  <title>The title</title>
</head>
```

Headings: HTML headings are defined with the <h1> to <h6> tags:

```
<h1>Heading1</h1>
<h2>Heading2</h2>
<h3>Heading3</h3>
<h4>Heading4</h4>
<h5>Heading5</h5>
<h6>Heading6</h6>
```

Paragraphs:

```
<p>Paragraph 1</p>  <p>Paragraph 2</p>
```

Line breaks:
. The difference between <br> and <p> is that 'br' breaks a line without altering the semantic structure of the page, whereas 'p' sections the page into paragraphs. Note also that 'br' is an *empty element* in that, while it may have attributes, it can take no content or end tag.

```
<p>This <br> is a paragraph <br> with <br> line breaks</p>
```

Comments:

```
<!-- Explanation here -->
```

Comments can help understanding of the markup and do not display in the webpage.

There are several types of markup elements used in HTML.

- **Structural** markup describes the purpose of text. For example, `<h2>Golf</h2>` establishes "Golf" as a second-level heading, which would be rendered in a browser in a manner similar to the "HTML markup" title at the start of this section. Structural markup does not denote any specific rendering, but most web browsers have default styles for element formatting. Text may be further styled with Cascading Style Sheets (CSS).
- **Presentational** markup describes the appearance of the text, regardless of its purpose. For example `<b>boldface</b>` indicates that visual output devices should render "boldface" in bold text, but gives little indication what devices which are unable to do this (such as aural devices that read the text aloud) should do. In the case of both `<b>bold</b>` and `<i>italic</i>`, there are other elements that may have equivalent visual renderings but which are more semantic in nature, such as `<strong>strong text</strong>` and `<em>emphasis text</em>` respectively. It is easier to see how an aural user agent should interpret the latter two elements. However, they are not equivalent to their presentational counterparts: it would be undesirable for a screen-reader to emphasize the name of a book, for instance, but on a screen such a name would be italicized. Most presentational markup elements have become deprecated under the HTML 4.0 specification, in favor of CSS based styling.
- **Hypertext** markup makes parts of a document into links to other documents. An anchor element creates a hyperlink in the document with the `href` attribute set to the link URL. For example, the HTML markup,

Attributes

Most of the attributes of an element are name-value pairs, separated by "=" and written within the start tag of an element after the element's name. The value may be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML). Leaving attribute values unquoted is considered unsafe. In contrast with name-value pair attributes, there are some attributes that affect the element simply by their presence in the start tag of the element, like the `ismap` attribute for the `img` element.

There are several common attributes that may appear in many elements:

- The `id` attribute provides a document-wide unique identifier for an element. This is used to identify the element so that stylesheets can alter its presentational properties, and scripts may alter, animate or delete its contents or presentation. Appended to the URL of the page, it provides a globally unique identifier for the element, typically a sub-section of the page.
- The `class` attribute provides a way of classifying similar elements. This can be used for semantic or presentation purposes. For example, an HTML document might semantically use the designation `class="notation"` to indicate that all elements with this class value are subordinate to the main text of the document. In presentation, such elements might be gathered together and presented as footnotes on a page instead of appearing in the place where they occur in the HTML source. Class attributes are used semantically in microformats. Multiple class values may

be specified; for example `class="notation important"` puts the element into both the 'notation' and the 'important' classes.
- An author may use the `style` attribute to assign presentational properties to a particular element. It is considered better practice to use an element's `id` or `class` attributes to select the element from within a stylesheet, though sometimes this can be too cumbersome for a simple, specific, or ad hoc styling.
- The `title` attribute is used to attach subtextual explanation to an element. In most browsers this attribute is displayed as a tooltip.
- The `lang` attribute identifies the natural language of the element's contents, which may be different from that of the rest of the document. For example, in an English-language document:
- `  <p>Oh well, <span lang="fr">c'est la vie</span>, as they say in France.</p>`

The abbreviation element, `abbr`, can be used to demonstrate some of these attributes:

```
<abbr id="anId" class="jargon" style="color:purple;" title="Hypertext
Markup Language">HTML</abbr>
```

This example displays as HTML; in most browsers, pointing the cursor at the abbreviation should display the title text "Hypertext Markup Language."

Most elements also take the language-related attribute `dir` to specify text direction, such as with "rtl" for right-to-left text in, for example, Arabic, Persian or Hebrew.

Character and entity references

As of version 4.0, HTML defines a set of 252 character entity references and a set of 1,114,050 numeric character references, both of which allow individual characters to be written via simple markup, rather than literally. A literal character and its markup counterpart are considered equivalent and are rendered identically.

The ability to "escape" characters in this way allows for the characters < and `&` (when written as `<` and `&`, respectively) to be interpreted as character data, rather than markup. For example, a literal < normally indicates the start of a tag, and `&` normally indicates the start of a character entity reference or numeric character reference; writing it as `&` or `&#x26;` or `&#38;` allows `&` to be included in the content of an element or in the value of an attribute. The double-quote character (`"`), when used to quote an attribute value, must also be escaped as `&quot;` or `&#x22;` or `&#34;` when it appears within the attribute value itself. Equivalently, the single-quote character (`'`), when used to quote an attribute value, must also be escaped as `&#x27;` or `&#39;` (not as `&apos;` except in XHTML documents) when it appears within the attribute value itself. If document authors overlook the need to escape such characters, some browsers can be very forgiving and try to use context to guess their intent. The result is still invalid markup, which makes the document less accessible to other browsers and to other user agents that may try to parse the document for search and indexing purposes for example.

Escaping also allows for characters that are not easily typed, or that are not available in the document's character encoding, to be represented within element and attribute content. For example, the acute-accented e (é), a character typically found only on Western European keyboards, can be written in any HTML document as the entity reference `&eacute;` or as the numeric references `&#233;` or `&#xE9;`, using characters that are available on all keyboards and are supported in all character encodings. Unicode character encodings such as UTF-8 are compatible with all modern browsers and allow direct access to almost all the characters of the world's writing systems.

Data types

HTML defines several data types for element content, such as script data and stylesheet data, and a plethora of types for attribute values, including IDs, names, URIs, numbers, units of length, languages, media descriptors, colors, character encodings, dates and times, and so on. All of these data types are specializations of character data.

Document type declaration

HTML documents are required to start with a Document Type Declaration (informally, a "doctype"). In browsers, the doctype helps to define the rendering mode—particularly whether to use quirks mode.

The original purpose of the doctype was to enable parsing and validation of HTML documents by SGML tools based on the Document Type Definition (DTD). The DTD to which the DOCTYPE refers contains a machine-readable grammar specifying the permitted and prohibited content for a document conforming to such a DTD. Browsers, on the other hand, do not implement HTML as an application of SGML and by consequence do not read the DTD. HTML5 does not define a DTD, because of the technology's inherent limitations, so in HTML5 the doctype declaration, `<!doctype html>`, does not refer to a DTD.

An example of an HTML 4 doctype is

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

This declaration references the DTD for the 'strict' version of HTML 4.01. SGML-based validators read the DTD in order to properly parse the document and to perform validation. In modern browsers, a valid doctype activates standards mode as opposed to quirks mode.

In addition, HTML 4.01 provides Transitional and Frameset DTDs, as explained below.

Semantic HTML

Semantic HTML is a way of writing HTML that emphasizes the meaning of the encoded information over its presentation (look). HTML has included semantic markup from its inception, but has also included presentational markup such as `<font>`, `<i>` and

 tags. There are also the semantically neutral span and div tags. Since the late 1990s when Cascading Style Sheets were beginning to work in most browsers, web authors have been encouraged to avoid the use of presentational HTML markup with a view to the separation of presentation and content.

In a 2001 discussion of the Semantic Web, Tim Berners-Lee and others gave examples of ways in which intelligent software 'agents' may one day automatically trawl the Web and find, filter and correlate previously unrelated, published facts for the benefit of human users. Such agents are not commonplace even now, but some of the ideas of Web 2.0, mashups and price comparison websites may be coming close. The main difference between these web application hybrids and Berners-Lee's semantic agents lies in the fact that the current aggregation and hybridisation of information is usually designed in by web developers, who already know the web locations and the API semantics of the specific data they wish to mash, compare and combine.

An important type of web agent that does trawl and read web pages automatically, without prior knowledge of what it might find, is the Web crawler or search-engine spider. These software agents are dependent on the semantic clarity of web pages they find as they use various techniques and algorithms to read and index millions of web pages a day and provide web users with search facilities without which the World Wide Web would be only a fraction of its current usefulness.

In order for search-engine spiders to be able to rate the significance of pieces of text they find in HTML documents, and also for those creating mashups and other hybrids as well as for more automated agents as they are developed, the semantic structures that exist in HTML need to be widely and uniformly applied to bring out the meaning of published text.

Presentational markup tags are deprecated in current HTML and XHTML recommendations and are illegal in HTML5.

Good semantic HTML also improves the accessibility of web documents. For example, when a screen reader or audio browser can correctly ascertain the structure of a document, it will not waste the visually impaired user's time by reading out repeated or irrelevant information when it has been marked up correctly.

Delivery

HTML documents can be delivered by the same means as any other computer file. However, they are most often delivered either by HTTP from a web server or by email.

HTTP

The World Wide Web is composed primarily of HTML documents transmitted from web servers to web browsers using the Hypertext Transfer Protocol (HTTP). However, HTTP is used to serve images, sound, and other content, in addition to HTML. To allow the Web browser to know how to handle each document it receives, other information is

transmitted along with the document. This meta data usually includes the MIME type (e.g. `text/html` or `application/xhtml+xml`) and the character encoding.

In modern browsers, the MIME type that is sent with the HTML document may affect how the document is initially interpreted. A document sent with the XHTML MIME type is expected to be well-formed XML; syntax errors may cause the browser to fail to render it. The same document sent with the HTML MIME type might be displayed successfully, since some browsers are more lenient with HTML.

The W3C recommendations state that XHTML 1.0 documents that follow guidelines set forth in the recommendation's Appendix C may be labeled with either MIME Type. The current XHTML 1.1 Working Draft also states that XHTML 1.1 documents should be labeled with either MIME type.

HTML e-mail

Most graphical email clients allow the use of a subset of HTML (often ill-defined) to provide formatting and semantic markup not available with plain text. This may include typographic information like coloured headings, emphasized and quoted text, inline images and diagrams. Many such clients include both a GUI editor for composing HTML e-mail messages and a rendering engine for displaying them. Use of HTML in e-mail is controversial because of compatibility issues, because it can help disguise phishing attacks, because it can confuse spam filters and because the message size is larger than plain text.

Naming conventions

The most common filename extension for files containing HTML is `.html`. A common abbreviation of this is `.htm`, which originated because some early operating systems and file systems, such as DOS and FAT, limited file extensions to three letters.

HTML Application

An HTML Application (HTA; file extension ".hta") is a Microsoft Windows application that uses HTML and Dynamic HTML in a browser to provide the application's graphical interface. A regular HTML file is confined to the security model of the web browser, communicating only to web servers and manipulating only webpage objects and site cookies. An HTA runs as a fully trusted application and therefore has more privileges, like creation/editing/removal of files and Windows Registry entries. Because they operate outside the browser's security model, HTAs cannot be executed via HTTP, but must be downloaded (just like an EXE file) and executed from local file system.

Current variations

> HTML is precisely what we were trying to PREVENT— ever-breaking links, links going outward only, quotes you can't follow to their origins, no version management, no rights management.
>
> Ted Nelson

Since its inception, HTML and its associated protocols gained acceptance relatively quickly. However, no clear standards existed in the early years of the language. Though its creators originally conceived of HTML as a semantic language devoid of presentation details, practical uses pushed many presentational elements and attributes into the language, driven largely by the various browser vendors. The latest standards surrounding HTML reflect efforts to overcome the sometimes chaotic development of the language and to create a rational foundation for building both meaningful and well-presented documents. To return HTML to its role as a semantic language, the W3C has developed style languages such as CSS and XSL to shoulder the burden of presentation. In conjunction, the HTML specification has slowly reined in the presentational elements.

There are two axes differentiating various variations of HTML as currently specified: SGML-based HTML versus XML-based HTML (referred to as XHTML) on one axis, and strict versus transitional (loose) versus frameset on the other axis.

SGML-based versus XML-based HTML

One difference in the latest HTML specifications lies in the distinction between the SGML-based specification and the XML-based specification. The XML-based specification is usually called XHTML to distinguish it clearly from the more traditional definition. However, the root element name continues to be 'html' even in the XHTML-specified HTML. The W3C intended XHTML 1.0 to be identical to HTML 4.01 except where limitations of XML over the more complex SGML require workarounds. Because XHTML and HTML are closely related, they are sometimes documented in parallel. In such circumstances, some authors conflate the two names as (X)HTML or X(HTML).

Like HTML 4.01, XHTML 1.0 has three sub-specifications: strict, transitional and frameset.

Aside from the different opening declarations for a document, the differences between an HTML 4.01 and XHTML 1.0 document—in each of the corresponding DTDs—are largely syntactic. The underlying syntax of HTML allows many shortcuts that XHTML does not, such as elements with optional opening or closing tags, and even EMPTY elements which must not have an end tag. By contrast, XHTML requires all elements to have an opening tag and a closing tag. XHTML, however, also introduces a new shortcut: an XHTML tag may be opened and closed within the same tag, by including a slash before the end of the tag like this: `<br/>`. The introduction of this shorthand, which is not used in the SGML declaration for HTML 4.01, may confuse earlier software unfamiliar with this new convention. A fix for this is to include a space before closing the tag, as such: `<br />`.

To understand the subtle differences between HTML and XHTML, consider the transformation of a valid and well-formed XHTML 1.0 document that adheres to Appendix C (see below) into a valid HTML 4.01 document. To make this translation requires the following steps:

1. **The language for an element should be specified with a `lang` attribute rather than the XHTML `xml:lang` attribute.** XHTML uses XML's built in language-defining functionality attribute.
2. **Remove the XML namespace (`xmlns=URI`).** HTML has no facilities for namespaces.
3. **Change the document type declaration** from XHTML 1.0 to HTML 4.01.
4. If present, **remove the XML declaration.** (Typically this is: `<?xml version="1.0" encoding="utf-8"?>`).
5. **Ensure that the document's MIME type is set to `text/html`.** For both HTML and XHTML, this comes from the HTTP `Content-Type` header sent by the server.
6. **Change the XML empty-element syntax to an HTML style empty element** (`<br/>` to `<br>`).

Those are the main changes necessary to translate a document from XHTML 1.0 to HTML 4.01. To translate from HTML to XHTML would also require the addition of any omitted opening or closing tags. Whether coding in HTML or XHTML it may just be best to always include the optional tags within an HTML document rather than remembering which tags can be omitted.

A well-formed XHTML document adheres to all the syntax requirements of XML. A valid document adheres to the content specification for XHTML, which describes the document structure.

The W3C recommends several conventions to ensure an easy migration between HTML and XHTML. The following steps can be applied to XHTML 1.0 documents only:

- Include both `xml:lang` and `lang` attributes on any elements assigning language.
- Use the empty-element syntax only for elements specified as empty in HTML.
- Include an extra space in empty-element tags: for example `<br />` instead of `<br/>`.
- Include explicit close tags for elements that permit content but are left empty (for example, `<div></div>`, not `<div />`).
- Omit the XML declaration.

By carefully following the W3C's compatibility guidelines, a user agent should be able to interpret the document equally as HTML or XHTML. For documents that are XHTML 1.0 and have been made compatible in this way, the W3C permits them to be served either as HTML (with a `text/html` MIME type), or as XHTML (with an `application/xhtml+xml` or `application/xml` MIME type). When delivered as XHTML, browsers should use an XML parser, which adheres strictly to the XML specifications for parsing the document's contents.

Transitional versus strict

HTML 4 defined three different versions of the language: Strict, Transitional (once called Loose) and Frameset. The Strict version is intended for new documents and is considered best practice, while the Transitional and Frameset versions were developed to make it

easier to transition documents that conformed to older HTML specification or didn't conform to any specification to a version of HTML 4. The Transitional and Frameset versions allow for presentational markup, which is omitted in the Strict version. Instead, cascading style sheets are encouraged to improve the presentation of HTML documents. Because XHTML 1 only defines an XML syntax for the language defined by HTML 4, the same differences apply to XHTML 1 as well. The Transitional version allows the following parts of the vocabulary, which are not included in the Strict version:

- **A looser content model**
  - Inline elements and plain text are allowed directly in: `body`, `blockquote`, `form`, `noscript` and `noframes`
- **Presentation related elements**
  - underline (`u`)(Deprecated. can confuse a visitor with a hyperlink.)
  - strike-through (`s`)
  - `center`(Deprecated. use CSS instead.)
  - `font`(Deprecated. use CSS instead.)
  - `basefont`(Deprecated. use CSS instead.)
- **Presentation related attributes**
  - `background`(Deprecated. use CSS instead.) and `bgcolor`(Deprecated. use CSS instead.) attributes for `body`(required element according to the W3C.) element.
  - `align`(Deprecated. use CSS instead.) attribute on `div`, `form`, paragraph (`p`) and heading (`h1`...`h6`) elements
  - `align`(Deprecated. use CSS instead.), `noshade`(Deprecated. use CSS instead.), `size`(Deprecated. use CSS instead.) and `width`(Deprecated. use CSS instead.) attributes on `hr` element
  - `align`(Deprecated. use CSS instead.), `border`, `vspace` and `hspace` attributes on `img` and `object`(caution: the `object` element is only supported in Internet Explorer(from the major browsers)) elements
  - `align`(Deprecated. use CSS instead.) attribute on `legend` and `caption` elements
  - `align`(Deprecated. use CSS instead.) and `bgcolor`(Deprecated. use CSS instead.) on `table` element
  - `nowrap`(Obsolete), `bgcolor`(Deprecated. use CSS instead.), `width`, `height` on `td` and `th` elements
  - `bgcolor`(Deprecated. use CSS instead.) attribute on `tr` element
  - `clear`(Obsolete) attribute on `br` element
  - `compact` attribute on `dl`, `dir` and `menu` elements
  - `type`(Deprecated. use CSS instead.), `compact`(Deprecated. use CSS instead.) and `start`(Deprecated. use CSS instead.) attributes on `ol` and `ul` elements
  - `type` and `value` attributes on `li` element
  - `width` attribute on `pre` element
- **Additional elements in Transitional specification**
  - `menu`(Deprecated. use CSS instead.) list (no substitute, though unordered list is recommended)
  - `dir`(Deprecated. use CSS instead.) list (no substitute, though unordered list is recommended)

    - `isindex`(Deprecated.) (element requires server-side support and is typically added to documents server-side, `form` and `input` elements can be used as a substitute)
    - `applet` (Deprecated. use the `object` element instead.)
- **The `language`(Obsolete) attribute on script element** (redundant with the `type` attribute).
- **Frame related entities**
    - `iframe`
    - `noframes`
    - `target`(Deprecated in the `map`, `link` and `form` elements.) attribute on `a`, client-side image-map (`map`), `link`, `form` and `base` elements

The Frameset version includes everything in the Transitional version, as well as the `frameset` element (used instead of `body`) and the `frame` element.

Frameset versus transitional

In addition to the above transitional differences, the frameset specifications (whether XHTML 1.0 or HTML 4.01) specifies a different content model, with `frameset` replacing `body`, that contains either `frame` elements, or optionally `noframes` with a `body`.

Summary of specification versions

As this list demonstrates, the loose versions of the specification are maintained for legacy support. However, contrary to popular misconceptions, the move to XHTML does not imply a removal of this legacy support. Rather the X in XML stands for extensible and the W3C is modularizing the entire specification and opening it up to independent extensions. The primary achievement in the move from XHTML 1.0 to XHTML 1.1 is the modularization of the entire specification. The strict version of HTML is deployed in XHTML 1.1 through a set of modular extensions to the base XHTML 1.1 specification. Likewise, someone looking for the loose (transitional) or frameset specifications will find similar extended XHTML 1.1 support (much of it is contained in the legacy or frame modules). The modularization also allows for separate features to develop on their own timetable. So for example, XHTML 1.1 will allow quicker migration to emerging XML standards such as MathML (a presentational and semantic math language based on XML) and XForms—a new highly advanced web-form technology to replace the existing HTML forms.

In summary, the HTML 4.01 specification primarily reined in all the various HTML implementations into a single clearly written specification based on SGML. XHTML 1.0, ported this specification, as is, to the new XML defined specification. Next, XHTML 1.1 takes advantage of the extensible nature of XML and modularizes the whole specification. XHTML 2.0 will be the first step in adding new features to the specification in a standards-body-based approach.

Hypertext features not in HTML

HTML lacks some of the features found in earlier hypertext systems, such as typed links, source tracking, fat links and others. Even some hypertext features that were in early versions of HTML have been ignored by most popular web browsers until recently, such as the link element and in-browser Web page editing.

WYSIWYG editors

There are some WYSIWYG editors (What You See Is What You Get), in which the user lays out everything as it is to appear in the HTML document using a graphical user interface (GUI), where the editor renders this as an HTML document, no longer requiring the author to have extensive knowledge of HTML.

The WYSIWYG editing model has been criticized, primarily because of the low quality of the generated code; there are voices advocating a change to the WYSIWYM model (What You See Is What You Mean).

WYSIWYG editors remains a controversial topic because of their perceived flaws such as:

- Relying mainly on layout as opposed to meaning, often using markup that does not convey the intended meaning but simply copies the layout.
- Often producing extremely verbose and redundant code that fails to make use of the cascading nature of HTML and CSS.
- Often producing ungrammatical markup often called tag soup.
- As a great deal of information of HTML documents is not in the layout, the model has been criticized for its 'what you see is all you get'-nature.

Nevertheless, since WYSIWYG editors offer convenience over hand-coded pages as well as not requiring the author to know the finer details of HTML, they still dominate web authoring.