Web Engineering and Web Development

Scott Buchanan

Abby Tucker

First Edition, 2012

ISBN 978-81-323-2002-9

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

Table of Contents

Chapter 1 - Web Application

Chapter 2 - Web Design

Chapter 3 - Web Page

Chapter 4 - World Wide Web

Chapter 5 - Web Service

Chapter 6 - Semantic Web

Chapter 7 - Search Engine Optimization

Chapter 8 - Hypertext Transfer Protocol

Chapter 9 - Web Development

Chapter 10 - Ajax (Programming) and Web Syndication

Chapter 11 - Web Application Framework

Chapter 12 - Web Analytics

Chapter 13 - Web Colors

Chapter 14 - Comet (Programming)

Chapter 15 - Mashup (Web Application Hybrid)

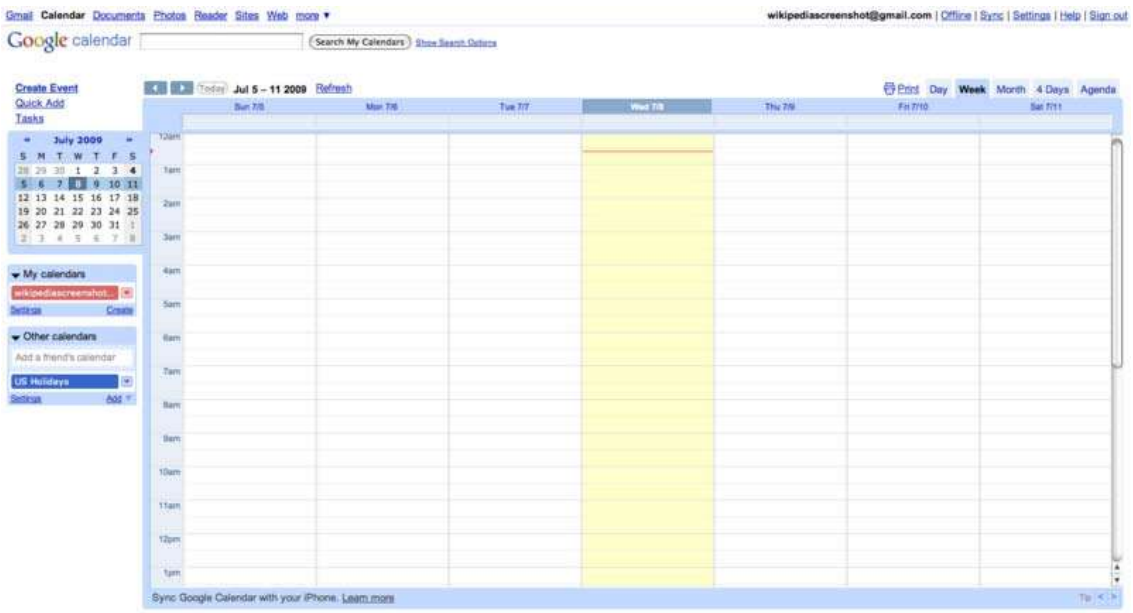
Chapter 16 - Microformat

Chapter 17 - Push Technology and Style Sheet (Web Development)

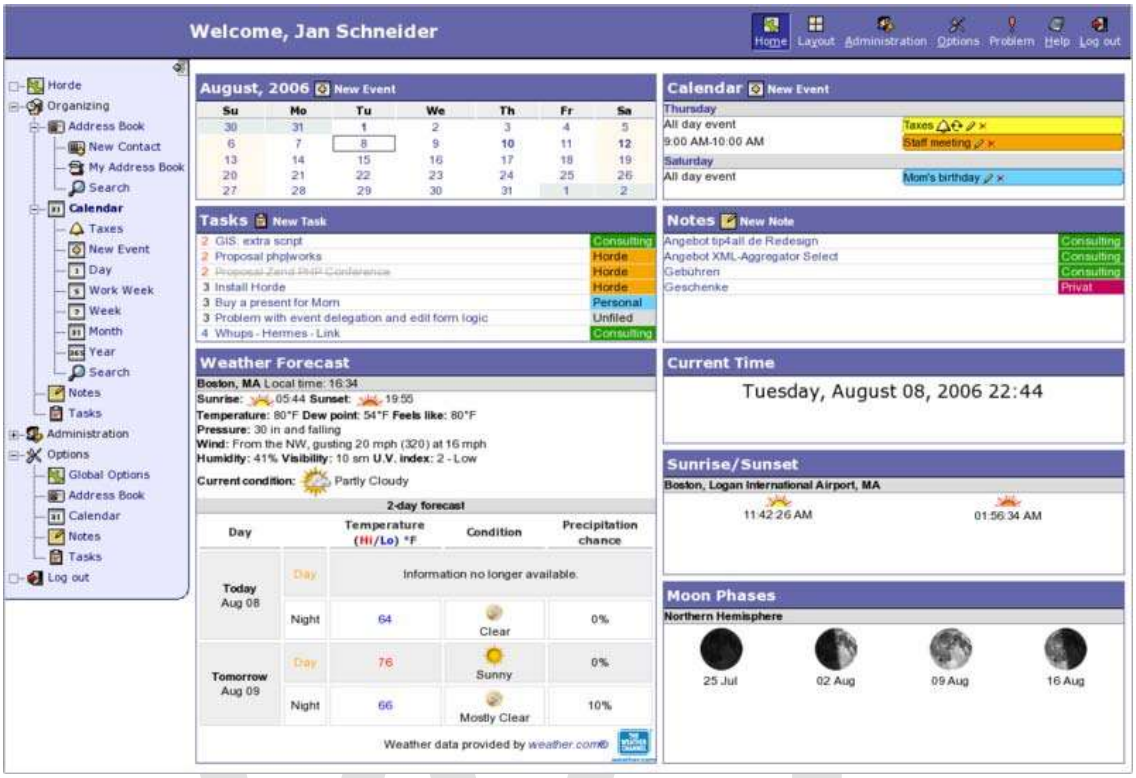
Chapter 18 - Web Workers and WebSockets

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.

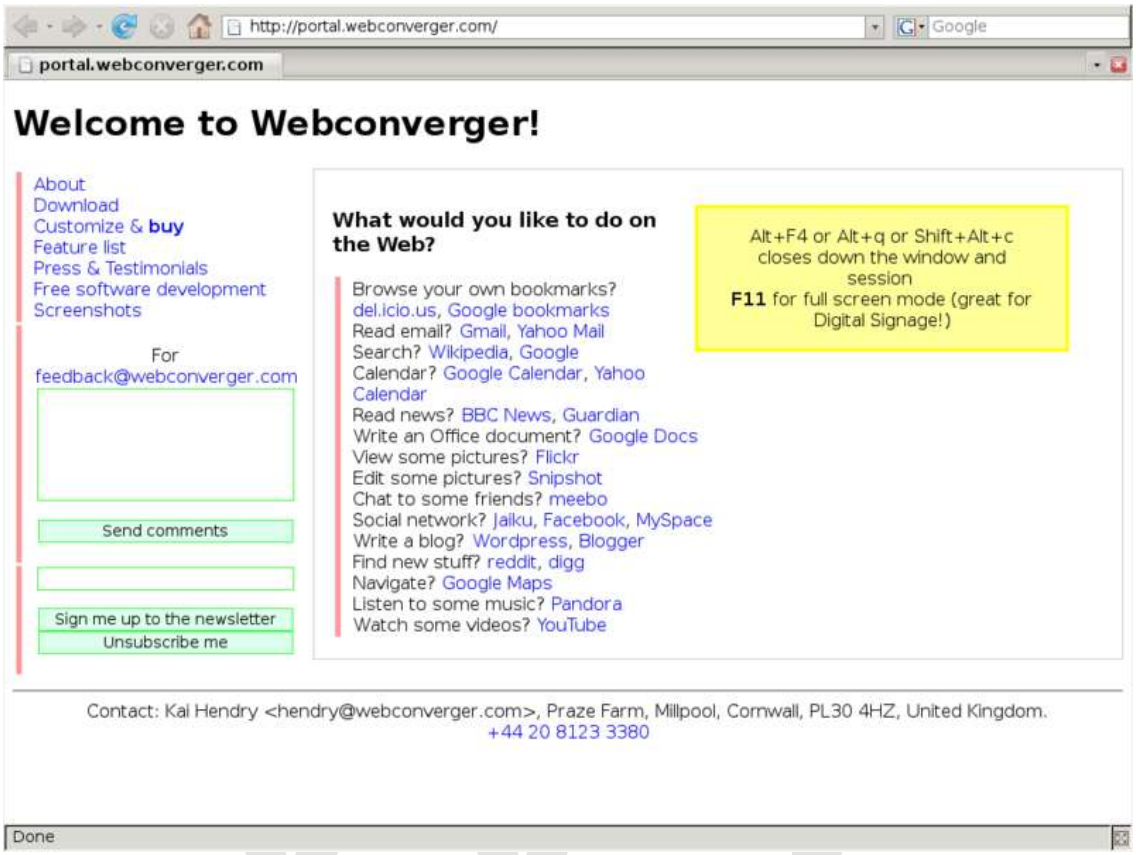
In 1995, Netscape introduced a client-side scripting language called JavaScript, which allowed programmers to add some dynamic elements to the user interface that ran on the client side. Until then, all the data had to be sent to the server for processing, and the results were delivered through static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

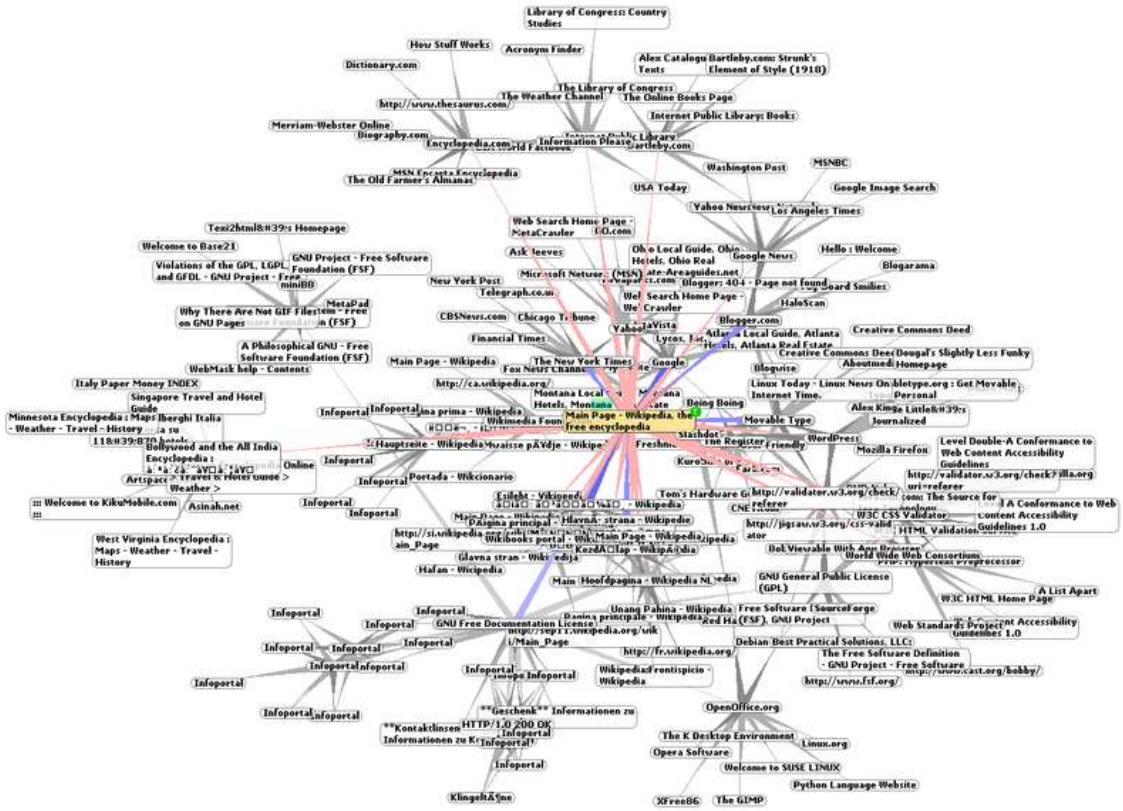
The computer receiving the HTTP request delivers it to Web server software listening for requests on port 80. If the web server can fulfill the request it sends an HTTP response back to the browser indicating success, which can be as simple as

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.



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.

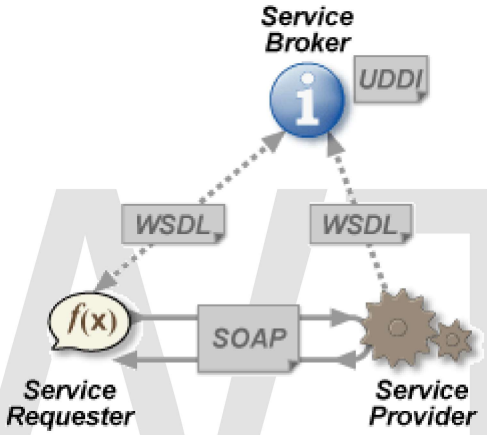
Caching

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."

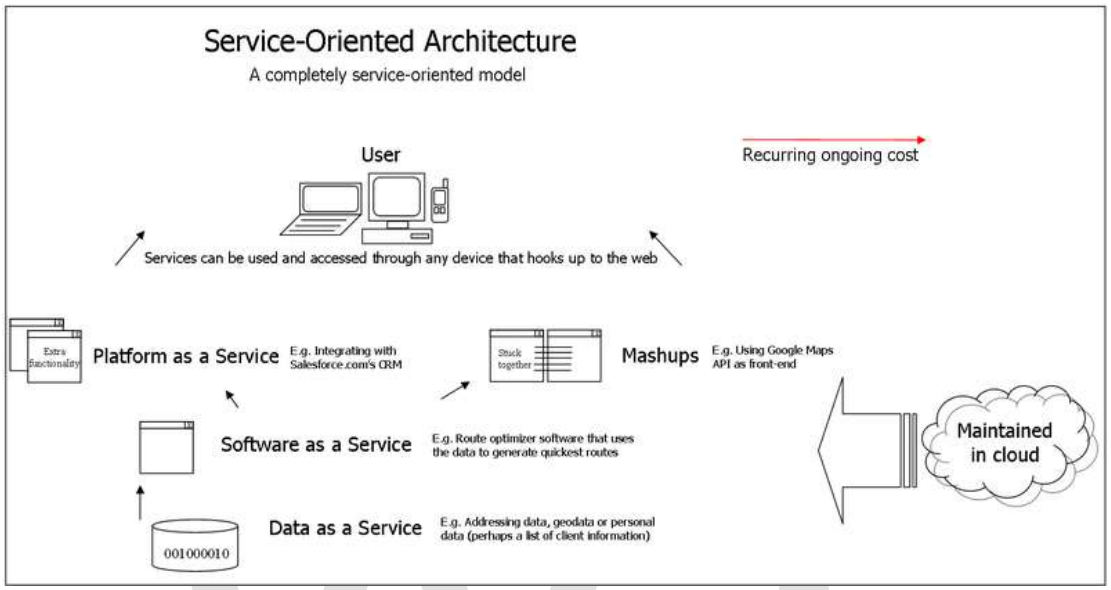
The W3C also states, "We can identify two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose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

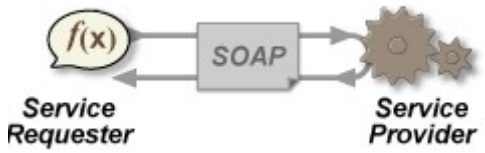
When used in the context of Web development, Web API is typically a defined set of Hypertext Transfer Protocol (HTTP) request messages along with a definition of the structure of response messages, usually expressed in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

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.

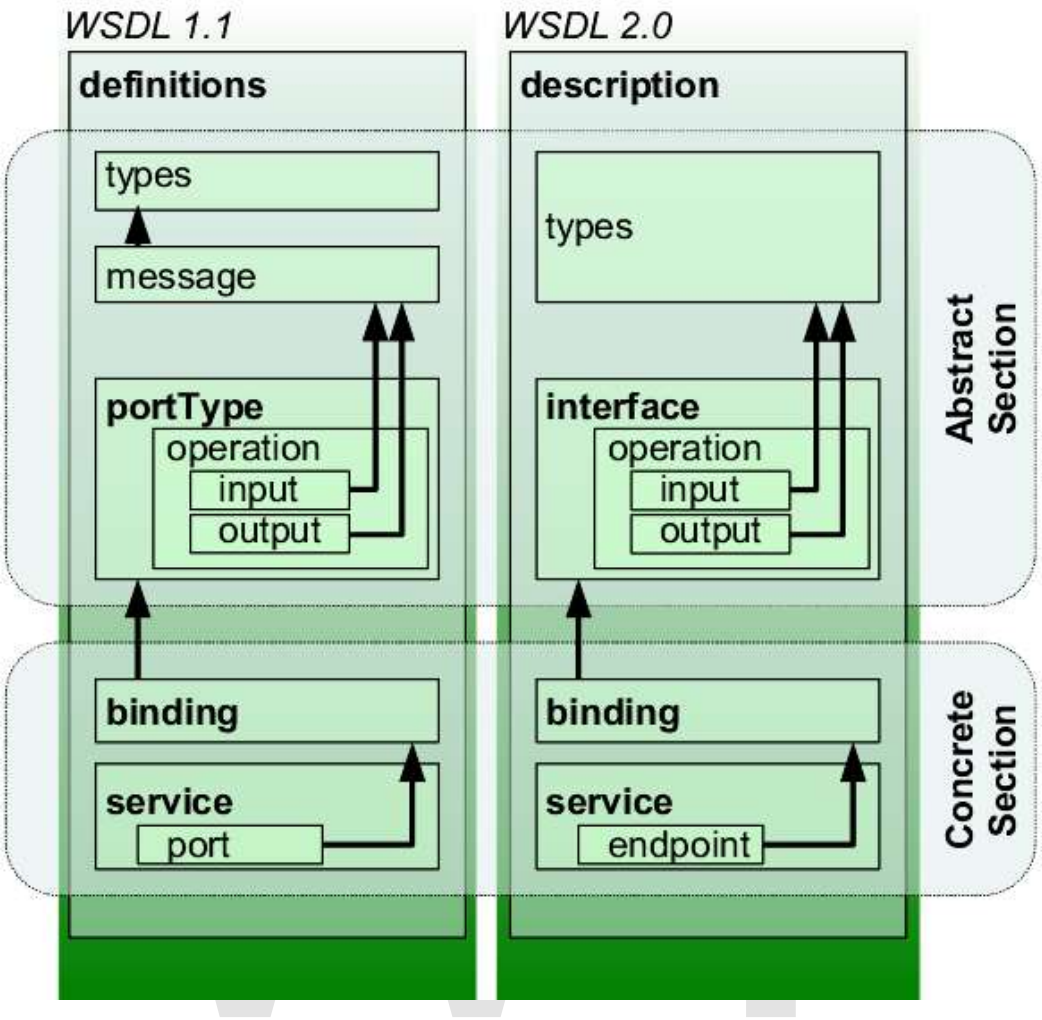
Other approaches with nearly the same functionality as RPC are Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) or Sun Microsystems's Java/Remote Method Invocation (RMI).

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.

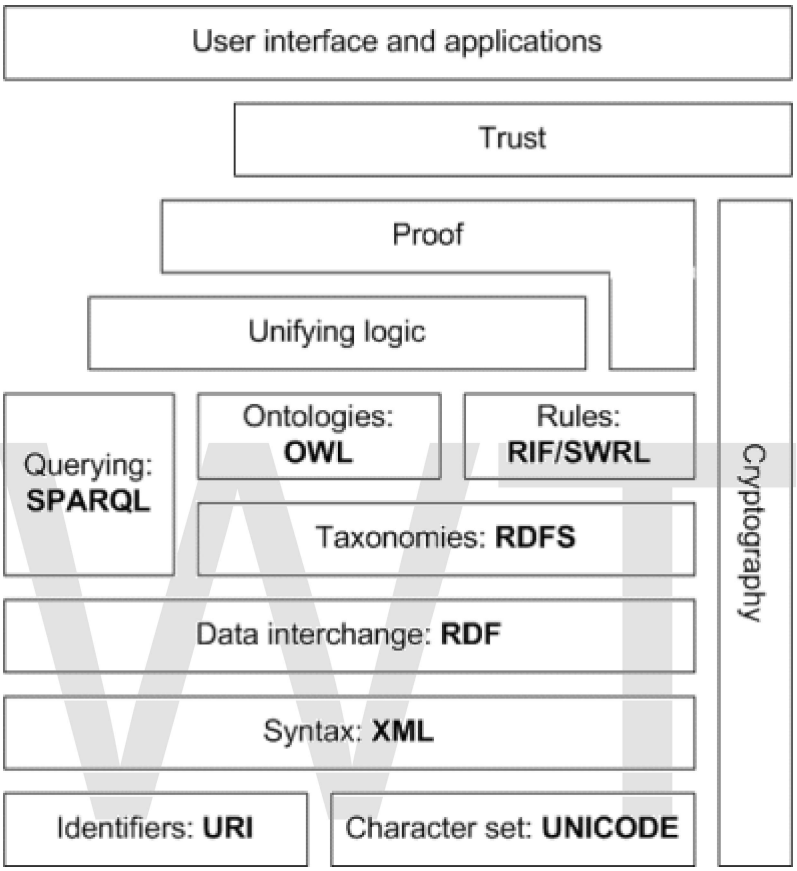
Need

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.

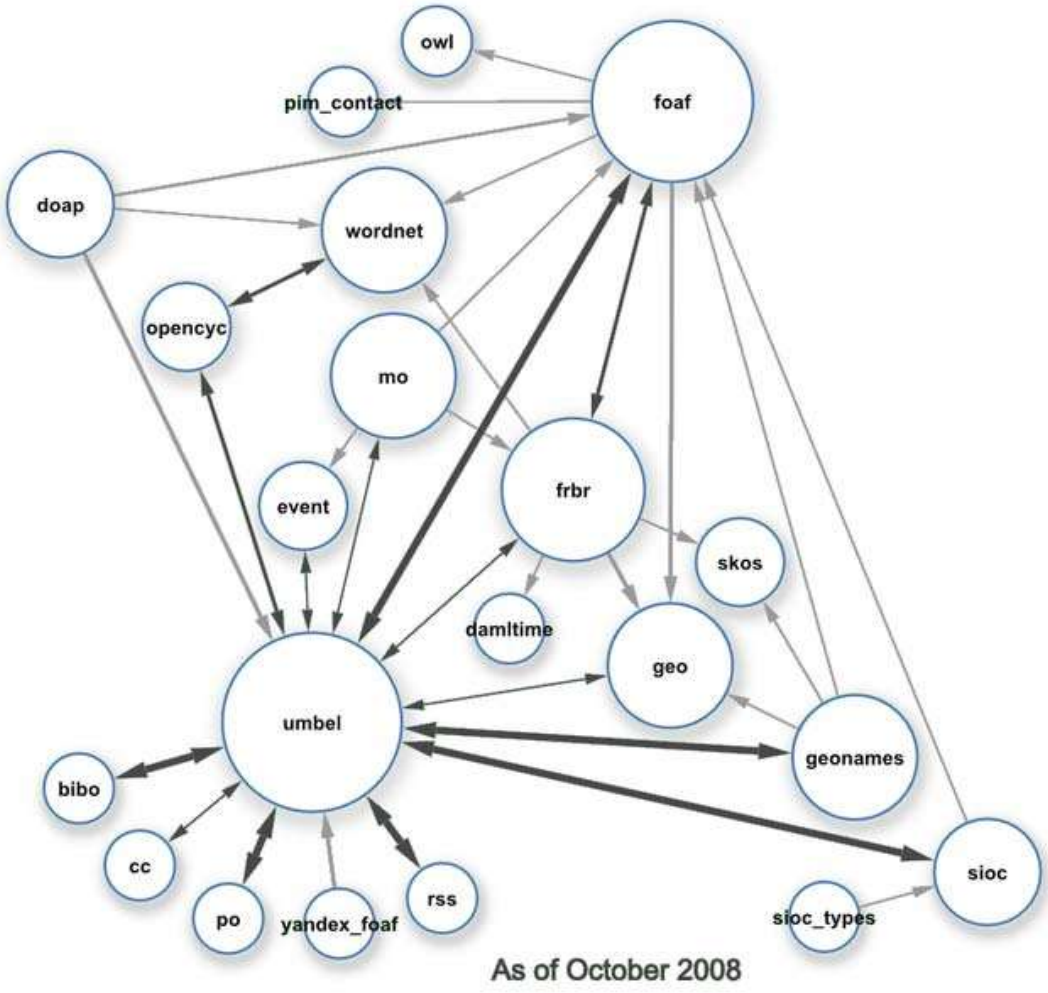
NextBio

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.

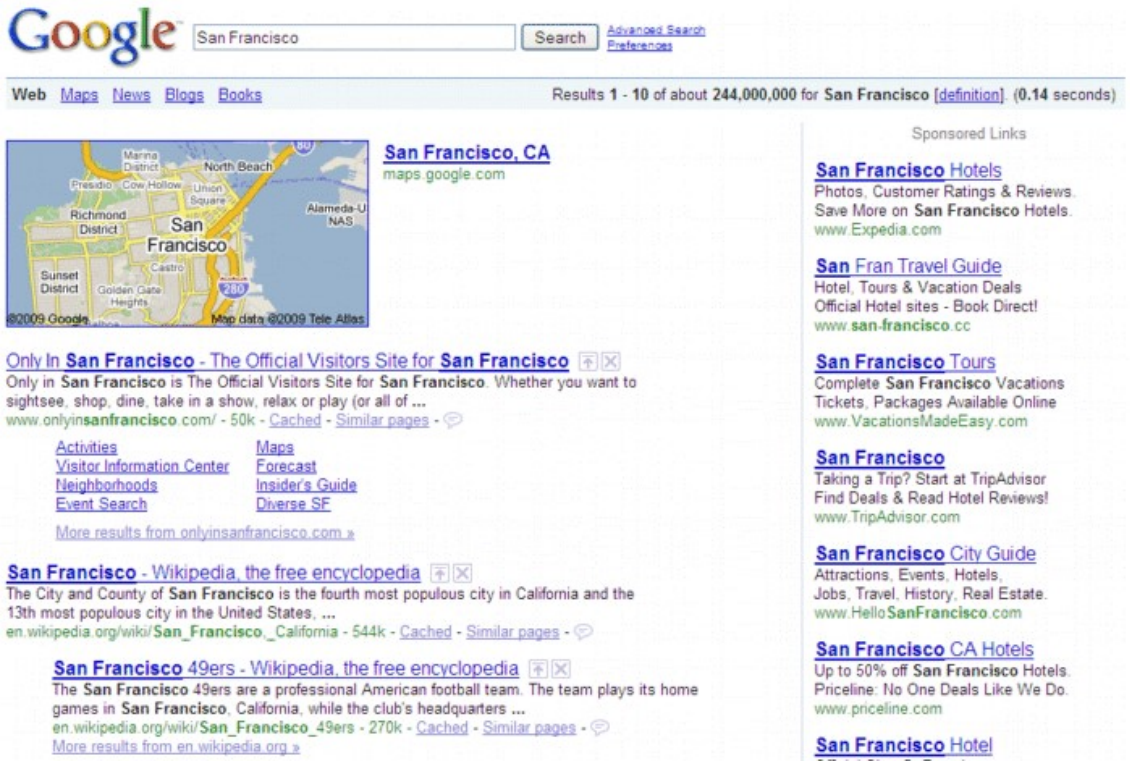
OpenPSI

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

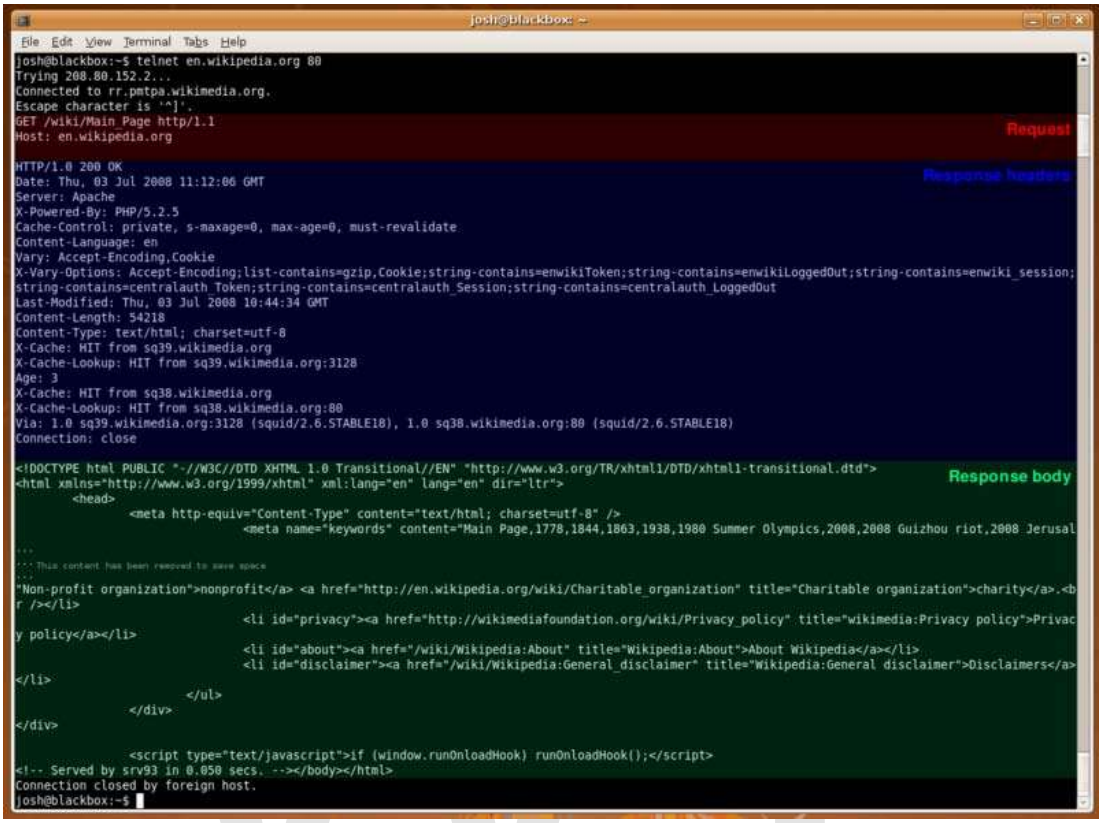
The request message consists of the following:

- 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 Development

Web development is a broad term for the work involved in developing a web site for the Internet (World Wide Web) or an intranet (a private network). This can include web design, web content development, client liaison, client-side/server-side scripting, web server and network security configuration, and e-commerce development. However, among web professionals, "web development" usually refers to the main non-design aspects of building web sites: writing markup and coding. Web development can range from developing the simplest static single page of plain text to the most complex web-based internet applications, electronic businesses, or social network services.

For larger organizations and businesses, web development teams can consist of hundreds of people (web developers). Smaller organizations may only require a single permanent or contracting webmaster, or secondary assignment to related job positions such as a graphic designer and/or information systems technician. Web development may be a collaborative effort between departments rather than the domain of a designated department.

Web development as an industry

Since the mid-1990s, web development has been one of the fastest growing industries in the world. In 1995 there were fewer than 1,000 web development companies in the United States, but by 2005 there were over 30,000 such companies in the U.S. alone. The growth of this industry is being pushed by large businesses wishing to sell products and services to their customers and to automate business workflow.

In addition, cost of Web site development and hosting has dropped dramatically during this time. Instead of costing ten thousands of dollars, as was the case for early websites, one can now develop a simple web site for less than a thousand dollars, depending on the complexity and amount of content. Smaller Web site development companies are now able to make web design accessible to both smaller companies and individuals further fueling the growth of the web development industry. As far as web development tools and platforms are concerned, there are many systems available to the public free of charge to aid in development. A popular example is the LAMP (Linux, Apache, MySQL, PHP) stack, which is usually distributed free of charge. This fact alone has manifested into many people around the globe setting up new Web sites daily and thus contributing to increase in web development popularity. Another contributing factor has been the rise of easy to use WYSIWYG web development software, most prominently Adobe Dreamweaver, Netbeans, WebDev, or Microsoft Expression Studio, Adobe Flex. Using

such software, virtually anyone can develop a Web page in a matter of minutes. Knowledge of HyperText Markup Language (HTML), or other programming languages is not required, but recommended for professional results.

The next generation of web development tools uses the strong growth in LAMP, Java Platform, Enterprise Edition technologies and Microsoft .NET technologies to provide the Web as a way to run applications online. Web developers now help to deliver applications as Web services which were traditionally only available as applications on a desk based computer.

Instead of running executable code on a local computer, users are interacting with online applications to create new content. This has created new methods in communication and allowed for many opportunities to decentralize information and media distribution. Users are now able to interact with applications from many locations, instead of being tied to a specific workstation for their application environment.

Examples of dramatic transformation in communication and commerce led by web development include e-commerce. Online auction sites such as eBay have changed the way consumers consume and purchase goods and services. Online resellers such as Amazon.com and Buy.com (among many, many others) have transformed the shopping and bargain hunting experience for many consumers. Another good example of transformative communication led by web development is the blog. Web applications such as WordPress and Movable Type have created easily implemented blog environments for individual Web sites. Open source content management systems such as Joomla!, Drupal, XOOPS, and TYPO3 and enterprise content management systems such as Alfresco have extended web development into new modes of interaction and communication.

In addition, web development has moved to a new phase of Internet communication. Computer web sites are no longer simply tools for work or commerce but used most for communication. Websites such as Facebook and Twitter provide users a platform to freely communicate. This new form of web communication is also changing e-commerce through the number of hits and online advertisement.

Typical Areas

Web Development can be split into many areas and a typical and basic web development hierarchy might consist of:

Client Side Coding

- Ajax Asynchronous JavaScript provides new methods of using JavaScript, and other languages to improve the user experience.
- Flash Adobe Flash Player is an ubiquitous browser plugin ready for RIAs. Flex 2 is also deployed to the Flash Player (version 9+).
- JavaScript Formally called ECMAScript, JavaScript is a ubiquitous client side platform for creating and delivering rich Web applications that can also run across a wide variety of devices.

- Microsoft Silverlight Microsoft's browser plugin that enables animation, vector graphics and high-definition video playback, programmed using XAML and .NET programming languages.
- REAL Studio Web Edition is a rapid application development environment for the web. The language is object oriented and is similar to both VB and Java. Applications are uniquely compiled to binary code.
- HTML5 and CSS3 Latest HTML proposed standard combined with the latest proposed standard for CSS natively supports much of the client-side functionality provided by other frameworks such as Flash and Silverlight

Server Side Coding

- ASP (Microsoft proprietary)
- CSP, Server-Side ANSI C
- ColdFusion (Adobe proprietary, formerly Macromedia, formerly Allaire)
- CGI and/or Perl (open source)
- Groovy (programming language) Grails (framework)
- Java, e.g. Java EE or WebObjects
- Lotus Domino
- PHP (open source)
- Python, e.g. Django (web framework) (open source)
- REAL Studio Web Edition
- Ruby, e.g. Ruby on Rails (open source)
- Smalltalk e.g. Seaside, AIDA/Web
- SSJS Server-Side JavaScript, e.g. Aptana Jaxer, Mozilla Rhino
- Websphere (IBM proprietary)
- .NET (Microsoft proprietary)

The World Wide Web has become a major delivery platform for web development 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.(1)(2) While web development can benefit from established practices from other related disciplines, it has certain distinguishing characteristics that demand special considerations. In recent years of web development 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.(3)(4) 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, web development encompasses new approaches, methodologies, tools, techniques, and guidelines to meet the unique requirements for web-based applications.

Client Side + Server Side

- Google Web Toolkit provides tools to create and maintain complex JavaScript front-end applications in Java.
- Pyjamas is a tool and framework for developing Ajax applications and Rich Internet Applications in python.
- Tersus is a platform for the development of rich web applications by visually defining user interface, client side behavior and server side processing. (open source)

However lesser known languages like Ruby and Python are often paired with database servers other than MySQL (the M in LAMP). Below are example of other databases currently in wide use on the web. For instance some developers prefer a LAPR(Linux/Apache/PostgreSQL/Ruby on Rails) setup for development.

Database Technology

- Apache Derby
- DB2 (IBM proprietary)
- Firebird
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- SQLite
- Sybase

In practice, many web developers will also have **interdisciplinary** skills / roles, including:

- Graphic design / web design
- Information architecture and copywriting/copyediting with web usability, accessibility and search engine optimization in mind
- Project management, QA and other aspects common to IT development in general

The above list is a simple website development hierarchy and can be extended to include all client side and server side aspects. It is still important to remember that web development is generally split up into client side coding, covering aspects such as the

layout and design, and server side coding, which covers the website's functionality and back end systems.

Looking at these items from an "umbrella approach", client side coding such as XHTML is executed and stored on a local client (in a web browser) whereas server side code is not available to a client and is executed on a web server which generates the appropriate XHTML which is then sent to the client. The nature of client side coding allows you to alter the HTML on a local client and refresh the pages with updated content (locally), web designers must bear in mind the importance and relevance to security with their server side scripts. If a server side script accepts content from a locally modified client side script, the web development of that page is poorly sanitized with relation to security.

Security Considerations

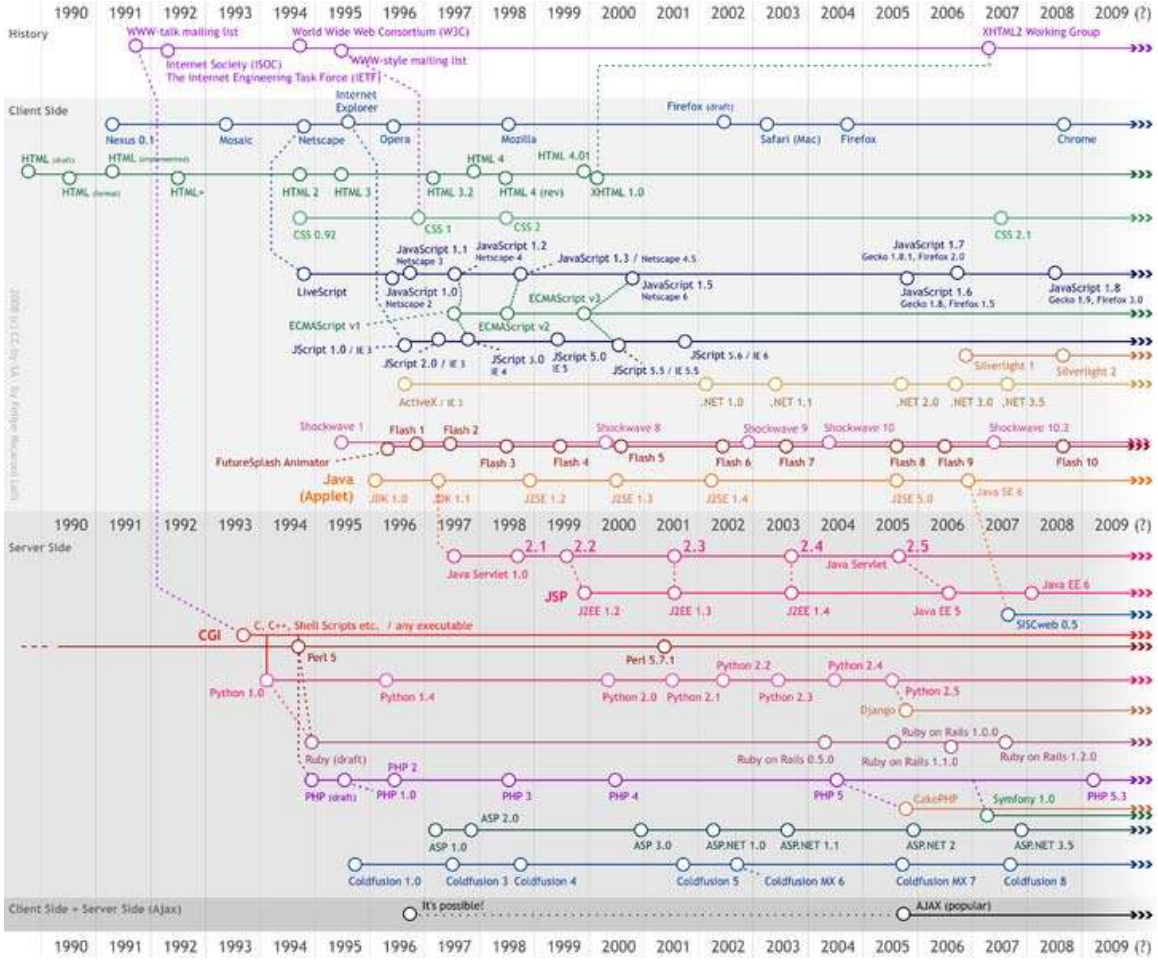
Web development takes into account many security considerations, such as data entry error checking through forms, filtering output, and encryption. Malicious practices such as SQL injection can be executed by users with ill intent yet with only primitive knowledge of web development as a whole. Scripts can be exploited to grant unauthorized access to malicious users trying to collect information such as email addresses, passwords and protected content like credit card numbers.

Some of this is dependent on the server environment (most commonly Apache or Microsoft IIS) on which the scripting language, such as PHP, Ruby, Python, Perl or ASP is running, and therefore is not necessarily down to the web developer themselves to maintain. However, stringent testing of web applications before public release is encouraged to prevent such exploits from occurring.

Keeping a web server safe from intrusion is often called *Server Port Hardening*. Many technologies come into play when keeping information on the internet safe when it is transmitted from one location to another. For instance Secure Socket Layer Encryption (SSL) Certificates are issued by certificate authorities to help prevent internet fraud. Many developers often employ different forms of encryption when transmitting and storing sensitive information. A basic understanding of information technology security concerns is often part of a web developer's knowledge.

Because new security holes are found in web applications even after testing and launch, security patch updates are frequent for widely used applications. It is often the job of web developers to keep applications up to date as security patches are released and new security concerns are discovered.

Timeline


1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 (?)
History
WWW-talk mailing list
World Wide Web Consortium (W3C)
XHTML2 Working Group
Internet Society (ISOC)
The Internet Engineering Task Force (IETF)
WWW-style mailing list
Client Side
Internet Explorer
Firefox (draft)
Nexus 0.1
Mosaic
Netscape
Opera
Mozilla
Safari (Mac)
Firefox
Chrome
HTML 4
HTML 4.01
HTML 2
HTML 3
HTML 3.2
HTML 4 (rev)
XHTML 1.0
CSS 0.92
CSS 1
CSS 2
CSS 2.1
JavaScript 1.1
JavaScript 1.2
JavaScript 1.3 / Netscape 4.5
JavaScript 1.7
LiveScript
JavaScript 1.0
JavaScript 1.5
JavaScript 1.6
JavaScript 1.8
ECMAScript v3
ECMAScript v1
ECMAScript v2
JScript 1.0 / IE 3
JScript 2.0 / IE 5
JScript 3.0
JScript 5.0
JScript 5.6 / IE 6
JScript 5.5 / IE 5.5
Silverlight 1
Silverlight 2
ActiveX / IE 3
.NET 1.0
.NET 1.1
.NET 2.0
.NET 3.0
.NET 3.5
Shockwave 1
Flash 1
Flash 2
Shockwave 8
Shockwave 9
Shockwave 10
Shockwave 10.2
FutureSplash Animator
Flash 3
Flash 4
Flash 5
Flash 6
Flash 7
Flash 8
Flash 9
Flash 10
Java (Applet)
J2SE 1.2
J2SE 1.3
J2SE 1.4
J2SE 5.0
Java SE 6
Server Side
2.1
2.2
2.3
2.4
2.5
Java Servlet 1.0
Java Servlet
JSP
J2EE 1.2
J2EE 1.3
J2EE 1.4
Java EE 5
Java EE 6
SISCweb 0.5
CGI
C, C++, Shell Scripts etc. / any executable
Perl 5
Perl 5.7.1
Python 1.0
Python 1.4
Python 2.0
Python 2.1
Python 2.2
Python 2.3
Python 2.4
Python 2.5
Django
Ruby (draft)
Ruby on Rails 0.5.0
Ruby on Rails 1.0.0
Ruby on Rails 1.1.0
Ruby on Rails 1.2.0
PHP (draft)
PHP 2
PHP 1.0
PHP 3
PHP 4
PHP 5
CakePHP
Symfony 1.0
PHP 5.3
ASP 2.0
ASP 1.0
ASP 3.0
ASP.NET 1.0
ASP.NET 1.1
ASP.NET 2
ASP.NET 3.5
Coldfusion 1.0
Coldfusion 3
Coldfusion 4
Coldfusion 5
Coldfusion MX 6
Coldfusion MX 7
Coldfusion 8
Client Side + Server Side (Ajax)
It's possible!
AJAX (popular)

Chapter 10

Ajax (Programming) and Web Syndication

Ajax (programming)

Ajax is a group of interrelated web development methods used on the client-side to create interactive web applications. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. Data is usually retrieved using the *XMLHttpRequest* object. Despite the name, the use of XML is not needed, and the requests need not be asynchronous.

Like DHTML and LAMP, Ajax is not one technology, but a group of technologies. Ajax uses a combination of HTML and CSS to mark up and style information. The DOM is accessed with JavaScript to dynamically display, and to allow the user to interact with the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

History

In the 1990s, most web sites were based on complete HTML pages; each user action required that the page be re-loaded from the server (or a new page loaded). This process is inefficient, as reflected by the user experience: all page content disappears then reappears, etc. Each time a page is reloaded due to a partial change, all of the content must be re-sent instead of only the changed information. This can place additional load on the server and use excessive bandwidth.

Asynchronous loading of content first became practical when Java applets were introduced in the first version of the Java language in 1995. These allow compiled client-side code to load data asynchronously from the web server after a web page is loaded. In 1996, Internet Explorer introduced the iframe element to HTML, which also enabled asynchronous loading. In 1999, Microsoft created the XMLHTTP ActiveX control in Internet Explorer 5, which was later adopted by Mozilla, Safari, Opera and other browsers as the XMLHttpRequest JavaScript object. Microsoft has adopted the native XMLHttpRequest model as of Internet Explorer 7, though the ActiveX version is still supported. The utility of background HTTP requests to the server and asynchronous web technologies remained fairly obscure until it started appearing in full scale online applications such as Outlook Web Access (2000) and Oddpost (2002), and later, Google made a wide deployment of Ajax with Gmail (2004) and Google Maps (2005).

The term *Ajax* was coined on February 18, 2005 by Jesse James Garrett in an article entitled *Ajax: A New Approach to Web Applications*.

On April 5, 2006 the World Wide Web Consortium (W3C) released the first draft specification for the XMLHttpRequest object in an attempt to create an official web standard.

Technologies

The term *Ajax* has come to represent a broad group of web technologies that can be used to implement a web application that communicates with a server in the background, without interfering with the current state of the page. In the chapter that coined the term Ajax, Jesse James Garrett explained that the following technologies are incorporated:

- HTML or XHTML and CSS for presentation
- the Document Object Model (DOM) for dynamic display of and interaction with data
- XML for the interchange of data, and XSLT for its manipulation
- the XMLHttpRequest object for asynchronous communication
- JavaScript to bring these technologies together

Since then, however, there have been a number of developments in the technologies used in an Ajax application, and the definition of the term Ajax. In particular, it has been noted that JavaScript is not the only client-side scripting language that can be used for implementing an Ajax application; other languages such as VBScript are also capable of the required functionality. (However, JavaScript is the most popular language for Ajax programming due to its inclusion in and compatibility with the majority of modern web browsers.) Also, XML is not required for data interchange and therefore XSLT is not required for the manipulation of data. JavaScript Object Notation (JSON) is often used as an alternative format for data interchange, although other formats such as preformatted HTML or plain text can also be used.

Drawbacks

- Pages dynamically created using successive Ajax requests do not automatically register themselves with the browser's history engine, so clicking the browser's "back" button may not return the browser to an earlier state of the Ajax-enabled page, but may instead return to the last full page visited before it. Workarounds include the use of invisible iframes to trigger changes in the browser's history and changing the URL fragment identifier (the part of a URL after the '#') when Ajax is run and monitoring it for changes.
- Dynamic web page updates also make it difficult to bookmark a particular state of the application. Solutions to this problem exist, many of which use the URL fragment identifier (the part of a URL after the '#') to keep track of, and allow returning to, the application in a given state.
- Depending on the nature of the Ajax application, dynamic page updates may interfere disruptively with user interactions, especially if working on an unstable

internet connection. For instance, editing a search field may trigger a query to the server for search completions, but the user may not know that a search completion popup is forthcoming, and if the internet connection is slow, the popup list may show up at an inconvenient time, when the user has already proceeded to do something else.

- Because most web crawlers do not execute JavaScript code, publicly indexable web applications should provide an alternative means of accessing the content that would normally be retrieved with Ajax, thereby allowing search engines to index it.
- Any user whose browser does not support JavaScript or XMLHttpRequest, or simply has this functionality disabled, will not be able to properly use pages which depend on Ajax. Similarly, devices such as mobile phones, PDAs, and screen readers may not have support for the required technologies. Screen readers that are able to use Ajax may still not be able to properly read the dynamically generated content. The only way to let the user carry out functionality is to fall back to non-JavaScript methods. This can be achieved by making sure links and forms can be resolved properly and do not rely solely on Ajax.
- The same origin policy prevents some Ajax techniques from being used across domains, although the W3C has a draft of the XMLHttpRequest object that would enable this functionality. Methods exist to sidestep this security feature by using a special Cross Domain Communications channel embedded as an iframe within a page, or by the use of JSONP.

- Ajax-powered interfaces may dramatically increase the number of user-generated requests to web servers and their back-ends (databases, or other). This can lead to longer response times and/or additional hardware needs.
- The asynchronous, callback-style of programming required can lead to complex code that is hard to maintain or debug.

- Ajax cannot easily be read by screen-reading technologies, such as JAWS, without hints built into the corresponding HTML based on WAI-ARIA standards. Screen-reading technologies are used by individuals with an impairment that hinders or prevents the ability to read the content on a screen.

Web syndication

Common web feed icon

Web syndication is a form of syndication in which website material is made available to multiple other sites. Most commonly, *web syndication* refers to making web feeds available from a site in order to provide other people with a summary or update of the website's recently added content (for example, the latest news or forum posts). The term can also be used to describe other kinds of licensing website content so that other websites can use it.

Motivation

Syndication benefits both the websites providing information and the websites displaying it. For the receiving site, content syndication is an effective way of adding greater depth and immediacy of information to its pages, making it more attractive to users. For the transmitting site, syndication drives exposure across numerous online platforms. This generates new traffic for the transmitting site — making syndication a relatively cheap, free and easy form of advertisement. However, as Search Engine Optimization has become an increasingly important topic among website owners and online marketers, content syndication has become a highly effective strategy for link building. Links embedded within the syndicated content are typically optimized around anchor terms that will point an optimized link back to the website that the content author is trying to promote. These links tell the algorithms of the search engines that the website being linked to is an authority for the keyword that is being used as the anchor text.

The prevalence of web syndication is also of note to online marketers, since web surfers are becoming increasingly wary of providing personal information for marketing materials (such as signing up for a newsletter) and expect the ability to subscribe to a feed instead. Although the format could be anything transported over HTTP, such as HTML or JavaScript, it is more commonly XML. The two main families of web syndication formats are RSS and Atom.

History

The basic idea of restructuring information about web sites goes back to as early as 1995, when Ramanathan V. Guha and others in Apple Computer's Advanced Technology Group developed the Meta Content Framework (MCF).

Large scale web syndication of content started in 1999 when Studio One Networks produced and distributed the first series of syndicated programs designed to be distributed on the Internet for its sponsor American Honda. Nowadays, many different types of content are syndicated on the Internet. Millions of online publishers, including newspapers, commercial websites and blogs, now publish their latest news headlines, product offers or blog postings in standard format news feed.

Web syndication as a commercial model

In addition to freely distributed material, some broadcasters and others use similar methods for the controlled placement of proprietary content on multiple partnering Internet destinations. In addition to web feeds, such commercial syndicators may use other methods to distribute their content such as Reuters, Associated Press, and All Headline News.

Such commercial web syndication borrows its business models from syndication in other media, such as Print, radio and television. Primarily, syndication arose in those other media so that content creators could reach a wider audience. In the case of radio, the United States Federal government proposed a syndicate in 1924 so that the nation's executives could quickly and efficiently reach the entire population. In the case of television, it is often said that "Syndication is where the real money is." Additionally, syndication accounts for the bulk of TV programming.

Commercial web syndication can be categorized in three ways:

- by **business models**
- by **types of content**
- by **methods for selecting distribution partners**

Commercial web syndication involves partnerships between content producers and distribution outlets. There are different structures of partnership agreements. One such structure is licensing content, in which distribution partners pay a fee to the content creators for the right to publish the content. Another structure is ad-supported content, in which publishers share revenues derived from advertising on syndicated content with that content's producer. A third structure is free, or barter syndication, in which no currency changes hands between publishers and content producers. This requires the content producers to generate revenue from another source, such as embedded advertising or subscriptions. Alternatively, they could distribute content without remuneration. Typically, those who create and distribute content for free are promotional entities, vanity publishers or government entities.

Types of content syndicated include RSS or Atom Feeds and full content. With RSS feeds, headlines, summaries, and sometimes a modified version of the original full content is displayed on users' feed readers. With full content, the entire content—which might be text, audio, video, applications/widgets or user-generated content—appears unaltered on the publisher's site.

There are two methods for selecting distribution partners. The content creator can hand-pick syndication partners based on specific criteria, such as the size or quality of their audiences. Alternatively, the content creator can allow publisher sites or users to "opt in" to carrying the content through an automated system. Some of these automated "content marketplace" systems involve careful screening of potential publishers by the content creator to ensure that the material does not end up in an inappropriate environment.

Just as syndication is a source of profit for TV producers and radio producers, it also functions to maximize profit for Internet content producers. As the Internet has increased in size it has become increasingly difficult for content producers to aggregate a sufficiently large audience to support the creation of high-quality content. Syndication enables content creators to amortize the cost of producing content by licensing it across multiple publishers or by maximizing distribution of advertising-supported content. However, a potential drawback for content creators is that they can lose control over the presentation of their content when they syndicate it to other parties.

Distribution partners benefit by receiving content either at a discounted price, or for free. One potential drawback for publishers, however, is that because the content is duplicated at other publisher sites, they cannot have an "exclusive" on the content.

For users, the fact that syndication enables the production and maintenance of content allows them to find and consume content on the Internet. One potential drawback for them is that they may run into duplicate content, which could be an annoyance.

Web syndication and e-commerce

Similar to syndication of proprietary content, web syndication has been used to distribute product content (feature descriptions, images, specifications etc.). Given that manufacturers are regarded as authorities and that most sales do not happen on manufacturer Web sites, best-in-class manufacturers take their best content and enable other retailers or dealers to publish the information on their sites. By syndicating content, a manufacturer is more likely to pass consistent and often comprehensive information to channel partners. Such web syndication has been shown to generate an increase in sales.

Web syndication has been increasingly used as a way to syndicate online news content to websites, too as part of Search Engine Optimisation techniques. Adding news content lets websites target specific keywords that could cost a fortune in PPC campaigns absolutely free. Also, by the very nature of writing about a topic - for example the UK housing market - you naturally cover other keyword variations in the same page, eg. house prices, mortgages, first-time buyers - all in one page: all keywords that someone interested in the UK housing market could type into a search engine also. Done on a daily basis, over time, news content will improve your search engine ranking considerably.

Chapter 11

Web Application Framework

A **web application framework** is a software framework that is designed to support the development of dynamic websites, web applications and web services. The framework aims to alleviate the overhead associated with common activities performed in Web development. For example, many frameworks provide libraries for database access, templating frameworks and session management, and they often promote code reuse.

History

As the design of the World Wide Web was not inherently dynamic, early hypertext consisted of hand-coded HTML that was published on web servers. Any modifications to published pages needed to be performed by the pages' author. To provide a dynamic web page that reflected user inputs, the Common Gateway Interface (CGI) standard was introduced for interfacing external applications with web servers. CGI could adversely affect server load, though, since each request had to start a separate process.

Programmers wanted tighter integration with the web server to enable high traffic web applications. The Apache HTTP Server, for example, supports modules that can extend the web server with arbitrary code executions (such as mod perl) or forward specific requests to a web server that can handle dynamic content (such as mod jk). Some web servers (such as Apache Tomcat) were specifically designed to handle dynamic content by executing code written in some languages, such as Java.

Around the same time, new languages were being developed specifically for use in the web, such as ColdFusion, PHP and Active Server Pages.

While the vast majority of languages available to programmers to use in creating dynamic web pages have libraries to help with common tasks, web applications often require specific libraries that are useful in web applications, such as creating HTML (for example, JavaServer Faces).

Eventually, mature, "full stack" frameworks appeared, that often gathered multiple libraries useful for web development into a single cohesive software stack for web developers to use. Examples of this include JavaEE (Servlets), WebObjects, OpenACS, Catalyst, Ruby on Rails, Django, and Zend Framework.

Architectures

Most web application frameworks are based on the MVC pattern. From an architecture perspective, there are generally five major types: request-based, component-based, hybrid, meta, and RIA-based.

Model view controller (MVC)

Many frameworks follow the Model View Controller (MVC) architectural pattern to separate the data model with business rules from user interface. This is generally considered a good practice as it modularizes code, promotes code reuse, and allows multiple interfaces to be applied.

Push-based vs. Pull-based

Most MVC frameworks follow a push-based architecture. These frameworks use actions that do the required processing, and then "push" the data to the view layer to render the results. Struts, Django, Ruby on Rails and Spring MVC are good examples of this architecture. An alternative to this is pull-based architecture, sometimes also called "component-based". These frameworks start with the view layer, which can then "pull" results from multiple controllers as needed. In this architecture, multiple controllers can be involved with a single view. Struts2, Lift, Tapestry, JBoss Seam, Wicket and Stripes are examples of pull-based architectures.

Content Management Systems

Some projects that have historically been termed content management systems have begun to take on the roles of higher-layer web application frameworks. For instance, Drupal's structure provides a minimal *core* whose function is extended through *modules* that provide functions generally associated with web application frameworks. However, it is debatable whether "management of content" is the primary value of such systems, especially when some, like SilverStripe, provide an object-oriented MVC framework. Add-on *modules* now enable these systems to function as full fledged applications beyond the scope of content management. They may provide functional APIs, functional frameworks, coding standards, and many of the functions traditionally associated with *Web application frameworks*.

Features

Web template system

Dynamic web pages usually consist of a static part (HTML) and a dynamic part, which is code that generates HTML. The code that generates the HTML can do this based on variables in a template, or on code. The text to be generated can come from a database, thereby making it possible to dramatically reduce the number of pages in a site.

Consider the example of a real estate agent with 500 houses for sale. In a static web site, the agent would have to create 500 pages in order to make the information available. In a dynamic website, the agent would simply connect the dynamic page to a database table of 500 records.

In a template, variables from the programming language can be inserted without using code, thereby losing the requirement of programming knowledge to make updates to the pages in a web site. A syntax is made available to distinguish between HTML and variables. E.g. in JSP the <c:out> tag is used to output variables, and in Smarty, {$variable} is used.

Many template engines do support limited logic tags, like IF and FOREACH. These are to be used only for decisions that need to be made for the presentation layer, in order to keep a clean separation from the business logic layer, or the M(odel) in the MVC pattern.

Caching

Web caching is the caching of web documents in order to reduce bandwidth usage, server load, and perceived "lag". A web cache stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met. Some application frameworks provide mechanisms for caching documents and bypassing various stages of the page's preparation, such as database access or template interpretation.

Security

Some web application frameworks come with authentication and authorization frameworks, that enable the web server to identify the users of the application, and restrict access to functions based on some defined criteria. Drupal is one example that provides role-based access to pages, and provides a web-based interface for creating users and assigning them roles.

Database access and mapping

Many web application frameworks create a unified API to a database backend, enabling web applications to work with a variety of databases with no code changes, and allowing programmers to work with higher-level concepts. For higher performance, database connections should be pooled as e.g. AOLserver does. Additionally, some object-oriented frameworks contain mapping tools to provide Object-Relational Mapping, which will map objects to tuples.

Other features web application frameworks may provide include transactional support and database migration tools.

URL mapping

A framework's URL mapping facility is the mechanism by which the framework interprets URLs. Some frameworks, such as Drupal and Django, match the provided

URL against pre-determined patterns using regular expressions, while some others use URL Rewriting to translate the provided URL into one that the underlying engine will recognize. Another technique is that of graph traversal such as used by Zope, where a URL is decomposed in steps that traverse an object graph (of models and views).

A URL mapping system that uses pattern matching or URL rewriting allows more "friendly" URLs to be used, increasing the simplicity of the site and allowing for better indexing by search engines. For example, a URL that ends with "/page.cgi?cat=science&topic=physics" could be changed to simply "/page/science/physics". This makes the URL easier to read and provides search engines with better information about the structural layout of the site. A graph traversal approach also tends to result in the creation of friendly URLs. A shorter URL such as "/page/science" tends to exist by default as that is simply a shorter form of the longer traversal to "/page/science/physics".

Ajax

Ajax, shorthand for "*Asynchronous JavaScript and XML*", is a web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is intended to increase the web page's interactivity, speed, and usability.

Due to the complexity of Ajax programming in Javascript, there are numerous Ajax frameworks that exclusively deal with Ajax support. Some Ajax frameworks are even embedded as a part of larger frameworks. For example, the Prototype JavaScript Framework is included in Ruby on Rails.

With the increased interest in developing "Web 2.0" Rich Media Applications, the complexity of programming directly in Ajax and Javascript has become so apparent that compiler technology has stepped in, to allow developers to code in high-level languages such as Java, Python and Ruby. The first of these compilers was Morfik followed by Google Web Toolkit, with ports to Python and Ruby in the form of Pyjamas and RubyJS following some time after. These compilers and their associated widget set libraries make the development of Rich Media Ajax Applications much more akin to that of developing Desktop applications.

Automatic configuration

Some frameworks minimize web application configuration through the use of introspection and/or following known conventions. For example, many Java frameworks use Hibernate as a persistence layer, which can generate a database schema at runtime capable of persisting the necessary information. This allows the application designer to design business objects without needing to explicitly define a database schema. Frameworks such as Ruby on Rails can also work in reverse, that is, define properties of model objects at runtime based on a database schema.

Web services

Some frameworks provide tools for creating and providing web services. These utilities may offer similar tools as the rest of the web application.

Chapter 12

Web Analytics

Web analytics is the measurement, collection, analysis and reporting of internet data for purposes of understanding and optimizing web usage.

Web analytics is not just a tool for measuring website traffic but can be used as a tool for business research and market research. Web analytics applications can also help companies measure the results of traditional print advertising campaigns. It helps one to estimate how the traffic to the website changed after the launch of a new advertising campaign. Web analytics provides data on the number of visitors, page views, etc. to gauge the traffic and popularity trends which helps doing the market research.

There are two categories of web analytics; *off-site* and *on-site* web analytics.

Off-site web analytics refers to web measurement and analysis regardless of whether you own or maintain a website. It includes the measurement of a website's *potential* audience (opportunity), share of voice (visibility), and buzz (comments) that is happening on the Internet as a whole.

On-site web analytics measure a visitor's journey once *on your website*. This includes its drivers and conversions; for example, which landing pages encourage people to make a purchase. On-site web analytics measures the performance of your website in a commercial context. This data is typically compared against key performance indicators for performance, and used to improve a web site or marketing campaign's audience response.

Historically, web analytics has referred to on-site visitor measurement. However in recent years this has blurred, mainly because vendors are producing tools that span both categories.

On-site web analytics technologies

Many different vendors provide on-site web analytics software and services. There are two main technological approaches to collecting the data. The first method, *logfile analysis*, reads the logfiles in which the web server records all its transactions. The second method, *page tagging*, uses JavaScript on each page to notify a third-party server when a page is rendered by a web browser. Both collect data that can be processed to produce web traffic reports.

In addition other data sources may also be added to augment the data. For example; e-mail response rates, direct mail campaign data, sales and lead information, user performance data such as click heat mapping, or other custom metrics as needed.

Web server logfile analysis

Web servers record some of their transactions in a logfile. It was soon realized that these logfiles could be read by a program to provide data on the popularity of the website. Thus arose web log analysis software.

In the early 1990s, web site statistics consisted primarily of counting the number of client requests (or *hits*) made to the web server. This was a reasonable method initially, since each web site often consisted of a single HTML file. However, with the introduction of images in HTML, and web sites that spanned multiple HTML files, this count became less useful. The first true commercial Log Analyzer was released by IPRO in 1994 .

Two units of measure were introduced in the mid 1990s to gauge more accurately the amount of human activity on web servers. These were *page views* and *visits* (or *sessions*). A *page view* was defined as a request made to the web server for a page, as opposed to a graphic, while a *visit* was defined as a sequence of requests from a uniquely identified client that expired after a certain amount of inactivity, usually 30 minutes. The page views and visits are still commonly displayed metrics, but are now considered rather rudimentary.

The emergence of search engine spiders and robots in the late 1990s, along with web proxies and dynamically assigned IP addresses for large companies and ISPs, made it more difficult to identify unique human visitors to a website. Log analyzers responded by tracking visits by cookies, and by ignoring requests from known spiders.

The extensive use of web caches also presented a problem for logfile analysis. If a person revisits a page, the second request will often be retrieved from the browser's cache, and so no request will be received by the web server. This means that the person's path through the site is lost. Caching can be defeated by configuring the web server, but this can result in degraded performance for the visitor to the website.

Page tagging

Concerns about the accuracy of logfile analysis in the presence of caching, and the desire to be able to perform web analytics as an outsourced service, led to the second data collection method, page tagging or 'Web bugs'.

In the mid 1990s, Web counters were commonly seen — these were images included in a web page that showed the number of times the image had been requested, which was an estimate of the number of visits to that page. In the late 1990s this concept evolved to include a small invisible image instead of a visible one, and, by using JavaScript, to pass along with the image request certain information about the page and the visitor. This information can then be processed remotely by a web analytics company, and extensive statistics generated.

The web analytics service also manages the process of assigning a cookie to the user, which can uniquely identify them during their visit and in subsequent visits. Cookie acceptance rates vary significantly between web sites and may affect the quality of data collected and reported.

Collecting web site data using a third-party data collection server (or even an in-house data collection server) requires an additional DNS look-up by the user's computer to determine the IP address of the collection server. On occasion, delays in completing a successful or failed DNS look-ups may result in data not being collected.

With the increasing popularity of Ajax-based solutions, an alternative to the use of an invisible image, is to implement a call back to the server from the rendered page. In this case, when the page is rendered on the web browser, a piece of Ajax code would call back to the server and pass information about the client that can then be aggregated by a web analytics company. This is in some ways flawed by browser restrictions on the servers which can be contacted with XmlHttpRequest objects.

Logfile analysis vs page tagging

Both logfile analysis programs and page tagging solutions are readily available to companies that wish to perform web analytics. In some cases, the same web analytics company will offer both approaches. The question then arises of which method a company should choose. There are advantages and disadvantages to each approach.

Advantages of logfile analysis

The main advantages of logfile analysis over page tagging are as follows:

- The web server normally already produces logfiles, so the raw data is already available. To collect data via page tagging requires changes to the website.
- The data is on the company's own servers, and is in a standard, rather than a proprietary, format. This makes it easy for a company to switch programs later, use several different programs, and analyze historical data with a new program.
- Page tagging solutions can involve vendor lock-in, but this problem can be avoided by using some type of include to embed the tracking code on each page from one file. It can then easily be changed for all pages simply by changing it in this one file, thus making it possible to easily add and remove analytics platforms without changing each page individually.
- Logfiles contain information on visits from search engine spiders. Although these should not be reported as part of the human activity, it is useful information for search engine optimization.
- Logfiles require no additional DNS Lookups. Thus there are no external server calls which can slow page load speeds, or result in uncounted page views.
- The web server reliably records every transaction it makes. Page tagging may not be able to record all transactions. Reasons include:
    - Page tagging relies on the visitors' browsers co-operating, which a certain proportion may not do (for example, if JavaScript is disabled, or a hosts file prohibits requests to certain servers).

- Tags may be omitted from pages either by oversight or between bouts of additional page tagging.
- It may not be possible to include tags in all pages. Examples include static content such as PDFs or application-generated dynamic pages where re-engineering the application to include tags is not an option.

Advantages of page tagging

The main advantages of page tagging over logfile analysis are as follows.

- Counting is activated by opening the page, not requesting it from the server. If a page is cached, it will not be counted by the server. Cached pages can account for up to one-third of all pageviews. Not counting cached pages seriously skews many site metrics. It is for this reason server-based log analysis is not considered suitable for analysis of human activity on websites.
- Data is gathered via a component ("tag") in the page, usually written in JavaScript, though Java can be used, and increasingly Flash is used. JQuery and AJAX can also be used in conjunction with a server-side scripting language (such as PHP) to manipulate and (usually) store it in a database, basically enabling complete control over the how data is represented.
- It is easier to add additional information to the tag, which can then be collected by the remote server. For example, information about the visitors' screen sizes, or the price of the goods they purchased, can be added in this way. With logfile analysis, information not normally collected by the web server can only be recorded by modifying the URL.
- Page tagging can report on events which do not involve a request to the web server, such as interactions within Flash movies, partial form completion, mouse events such as onClick, onMouseOver, onFocus, onBlur etc.
- The page tagging service manages the process of assigning cookies to visitors; with logfile analysis, the server has to be configured to do this.
- Page tagging is available to companies who do not have access to their own web servers.
- Lately page tagging has become a standard in web analytics.

Economic factors

Logfile analysis is almost always performed in-house. Page tagging can be performed in-house, but it is more often provided as a third-party service. The economic difference between these two models can also be a consideration for a company deciding which to purchase.

- Logfile analysis typically involves a one-off software purchase; however, some vendors are introducing maximum annual page views with additional costs to process additional information. In addition to commercial offerings, several open-source logfile analysis tools are available free of charge.
- For Logfile analysis you have to store and archive your own data, which often grows very large quickly. Although the cost of hardware to do this is minimal, the overhead for an IT department can be considerable. For example, if you run out of

disk space your database may start over-writing old entries which can often be irreparable.
- For Logfile analysis you need to maintain the software, including updates and security patches.
- Complex page tagging vendors charge a monthly fee based on volume i.e. number of pageviews per month collected.

Which solution is cheaper to implement depends on the amount of technical expertise within the company, the vendor chosen, the amount of activity seen on the web sites, the depth and type of information sought, and the number of distinct web sites needing statistics.

Regardless of the vendor solution or data collection method employed, the cost of web visitor analysis and interpretation should also be included. That is, the cost of turning raw data into actionable information. This can be from the use of third party consultants, the hiring of an experienced web analyst, or the training of a suitable in-house person. A cost-benefit analysis can then be performed. For example, what revenue increase or cost savings can be gained by analysing the web visitor data?

Hybrid methods

Some companies are now producing programs that collect data through both logfiles and page tagging. By using a hybrid method, they aim to produce more accurate statistics than either method on its own. The first Hybrid solution was produced in 1998 by Rufus Evison, who then spun the product out to create a company based upon the increased accuracy of hybrid methods.

Visitors Geolocation

With IP geolocation, it is possible to track visitors location. Using IP geolocation database or API, visitors can be geolocated to city, region or country level.

IP Intelligence, or Internet Protocol (IP) Intelligence, is a technology that maps the Internet and catalogues IP addresses by parameters such as geographic location (country, region, state, city and postcode), connection type, Internet Service Provider (ISP), proxy information, and more. The first generation of IP Intelligence was referred to as geotargeting or geolocation technology. This information is used by businesses for online audience segmentation in applications such online advertising, behavioral targeting, content localization (or website localization), digital rights management, personalization, online fraud detection, geographic rights management, localized search, enhanced analytics, global traffic management, and content distribution.

Click analytics

Click analytics is a special type of web analytics that gives special attention to clicks (Point-and-click).

Commonly, click analytics focuses on on-site analytics. An editor of a web site uses click analytics to determine the performance of his or her particular site, with regards to where the users of the site are clicking.

Also, click analytics may happen real-time or "unreal"-time, depending on the type of information sought. Typically, front-page editors on high-traffic news media sites will want to monitor their pages in real-time, to optimize the content. Editors, designers or other types of stakeholders may analyze clicks on a wider time frame to aid them assess performance of writers, design elements or advertisements etc.

Data about clicks may be gathered in at least two ways. Ideally, a click is "logged" when it occurs, and this method requires some functionality that picks up relevant information when the event occurs. Alternatively, one may institute the assumption that a page view is a result of a click, and therefore log a simulated click that lead to that page view.

Customer lifecycle analytics

Customer lifecycle analytics is a visitor-centric approach to measuring that falls under the umbrella of lifecycle marketing. Page views, clicks and other events (such as API calls, access to third-party services, etc.) are all tied to an individual visitor instead of being stored as separate data points. Customer lifecycle analytics attempts to connect all the data points into a marketing funnel that can offer insights into visitor behavior and website optimization.

Other methods

Other methods of data collection are sometimes used. Packet sniffing collects data by sniffing the network traffic passing between the web server and the outside world. Packet sniffing involves no changes to the web pages or web servers. Integrating web analytics into the web server software itself is also possible. Both these methods claim to provide better real-time data than other methods.

Key definitions

There are no globally agreed definitions within web analytics as the industry bodies have been trying to agree definitions that are useful and definitive for some time. The main bodies who have had input in this area have been JICWEBS (The Joint Industry Committee for Web Standards in the UK and Ireland), ABCe (Audit Bureau of Circulations electronic, UK and Europe), The WAA (Web Analytics Association, US) and to a lesser extent the IAB (Interactive Advertising Bureau). This does not prevent the following list from being a useful guide, suffering only slightly from ambiguity. Both the WAA and the ABCe provide more definitive lists for those who are declaring their statistics using the metrics defined by either.

- **Hit -** A request for a file from the web server. Available only in log analysis. The number of hits received by a website is frequently cited to assert its popularity, but this number is extremely misleading and dramatically over-estimates popularity. A single web-page typically consists of multiple (often dozens) of

discrete files, each of which is counted as a hit as the page is downloaded, so the number of hits is really an arbitrary number more reflective of the complexity of individual pages on the website than the website's actual popularity. The total number of visitors or page views provides a more realistic and accurate assessment of popularity.

- **Page view -** A request for a file whose type is defined as a page in log analysis. An occurrence of the script being run in page tagging. In log analysis, a single page view may generate multiple hits as all the resources required to view the page (images, .js and .css files) are also requested from the web server.
- **Visit / Session -** A visit is defined as a series of page requests from the same uniquely identified client with a time of no more than 30 minutes between each page request. A session is defined as a series of page requests from the same uniquely identified client with a time of no more than 30 minutes and no requests for pages from other domains intervening between page requests. In other words, a session ends when someone goes to another site, or 30 minutes elapse between pageviews, whichever comes first. A visit ends only after a 30 minute time delay. If someone leaves a site, then returns within 30 minutes, this will count as one visit but two sessions. In practice, most systems ignore sessions and many analysts use both terms for visits. Because time between pageviews is critical to the definition of visits and sessions, a single page view does not constitute a visit or a session (it is a "bounce").
- **First Visit / First Session -** A visit from a visitor who has not made any previous visits.
- **Visitor / Unique Visitor / Unique User -** The uniquely identified client generating requests on the web server (log analysis) or viewing pages (page tagging) within a defined time period (i.e. day, week or month). A Unique Visitor counts once within the timescale. A visitor can make multiple visits. Identification is made to the visitor's computer, not the person, usually via cookie and/or IP+User Agent. Thus the same person visiting from two different computers will count as two Unique Visitors. Increasingly visitors are uniquely identified by Flash LSO's (Local Shared Object), which are less susceptible to privacy enforcement.
- **Repeat Visitor -** A visitor that has made at least one previous visit. The period between the last and current visit is called visitor recency and is measured in days.
- **New Visitor -** A visitor that has not made any previous visits. This definition creates a certain amount of confusion, and is sometimes substituted with analysis of first visits.
- **Impression -** An impression is each time an advertisement loads on a user's screen. Anytime you see a banner, that is an impression.
- **Singletons -** The number of visits where only a single page is viewed. While not a useful metric in and of itself the number of singletons is indicative of various forms of Click fraud as well as being used to calculate bounce rate and in some cases to identify automatons bots.
- **Bounce Rate -** The percentage of visits where the visitor enters and exits at the same page without visiting any other pages on the site in between.
- **% Exit -** The percentage of users who exit from a page.
- **Visibility time -** The time a single page (or a blog, Ad Banner...) is viewed.

- **Session Duration -** Average amount of time that visitors spend on the site each time they visit. This metric can be complicated by the fact that analytics programs can not measure the length of the final page view.
- **Page View Duration / Time on Page -** Average amount of time that visitors spend on each page of the site. As with Session Duration, this metric is complicated by the fact that analytics programs can not measure the length of the final page view unless they record a page close event, such as onUnload().
- **Active Time / Engagement Time -** Average amount of time that visitors spend actually interacting with content on a web page, based on mouse moves, clicks, hovers and scrolls. Unlike Session Duration and Page View Duration / Time on Page, this metric **can** accurately measure the length of engagement in the final page view.
- **Page Depth / Page Views per Session -** Page Depth is the average number of page views a visitor consumes before ending their session. It is calculated by dividing total number of page views by total number of sessions and is also called Page Views per Session or PV/Session.
- **Frequency / Session per Unique -** Frequency measures how often visitors come to a website. It is calculated by dividing the total number of sessions (or visits) by the total number of unique visitors. Sometimes it is used to measure the loyalty of your audience.
- **Click path -** the sequence of hyperlinks one or more website visitors follows on a given site.
- **Click -** "refers to a single instance of a user following a hyperlink from one page in a site to another". Some use click analytics to analyze their web sites.
- **Site Overlay** is a technique in which graphical statistics are shown besides each link on the web page. These statistics represent the percentage of clicks on each link.

Common sources of confusion in web analytics

The hotel problem

The hotel problem is generally the first problem encountered by a user of web analytics. The term was first coined by Rufus Evison explaining the problem at one of the Emetrics Summits and has now gained popularity as a simple expression of the problem and its resolution.

The problem is that the unique visitors for each day in a month do not add up to the same total as the unique visitors for that month. This appears to an inexperienced user to be a problem in whatever analytics software they are using. In fact it is a simple property of the metric definitions.

The way to picture the situation is by imagining a hotel. The hotel has two rooms (Room A and Room B).

	Day 1	Day 2	Day 3	Total
Room A	John	John	Jane	2 Unique Users

Room B	Mark	Jane	Mark	2 Unique Users
Total	2	2	2	?

As the table shows, the hotel has two unique users each day over three days. The sum of the totals with respect to the days is therefore six.

During the period each room has had two unique users. The sum of the totals with respect to the rooms is therefore four.

Actually only three visitors have been in the hotel over this period. The problem is that a person who stays in a room for two nights will get counted twice if you count them once on each day, but is only counted once if you are looking at the total for the period. Any software for web analytics will sum these correctly for whatever time period, thus leading to the problem when a user tries to compare the totals.

New visitors + Repeat visitors unequal to total visitors

Another common misconception in web analytics is that the sum of the new visitors and the repeat visitors ought to be the total number of visitors. Again this becomes clear if the visitors are viewed as individuals on a small scale, but still causes a large number of complaints that analytics software cannot be working because of a failure to understand the metrics.

Here the culprit is the metric of a new visitor. There is really no such thing as a new visitor when you are considering a web site from an ongoing perspective. If a visitor makes their first visit on a given day and then returns to the web site on the same day they are both a new visitor and a repeat visitor for that day. So if we look at them as an individual which are they? The answer has to be both, so the definition of the metric is at fault.

A new visitor is not an individual; it is a fact of the web measurement. For this reason it is easiest to conceptualize the same facet as a first visit (or first session). This resolves the conflict and so removes the confusion. Nobody expects the number of first visits to add to the number of repeat visitors to give the total number of visitors. The metric will have the same number as the new visitors, but it is clearer that it will not add in this fashion.

On the day in question there was a first visit made by our chosen individual. There was also a repeat visit made by the same individual. The number of first visits and the number of repeat visits will add up to the total number of visits for that day.

Web analytics methods

Problems with cookies

Historically, vendors of page-tagging analytics solutions have used third-party cookies sent from the vendor's domain instead of the domain of the website being browsed.

Third-party cookies can handle visitors who cross multiple unrelated domains within the company's site, since the cookie is always handled by the vendor's servers.

However, third-party cookies in principle allow tracking an individual user across the sites of different companies, allowing the analytics vendor to collate the user's activity on sites where he provided personal information with his activity on other sites where he thought he was anonymous. Although web analytics companies deny doing this, other companies such as companies supplying banner ads have done so. Privacy concerns about cookies have therefore led a noticeable minority of users to block or delete third-party cookies. In 2005, some reports showed that about 28% of Internet users blocked third-party cookies and 22% deleted them at least once a month.

Most vendors of page tagging solutions have now moved to provide at least the option of using first-party cookies (cookies assigned from the client subdomain).

Another problem is cookie deletion. When web analytics depend on cookies to identify unique visitors, the statistics are dependent on a persistent cookie to hold a unique visitor ID. When users delete cookies, they usually delete both first- and third-party cookies. If this is done between interactions with the site, the user will appear as a first-time visitor at their next interaction point. Without a persistent and unique visitor id, conversions, click-stream analysis, and other metrics dependent on the activities of a unique visitor over time, cannot be accurate.

Cookies are used because IP addresses are not always unique to users and may be shared by large groups or proxies. Other methods of uniquely identifying a user are technically challenging and would limit the trackable audience or would be considered suspicious. Cookies are the selected option because they reach the lowest common denominator without using technologies regarded as spyware.

Unique landing pages vs referrals for campaign tracking

Tracking the amount of activity generated through advertising relationships with external web sites through the referrals reports available in most web analytics packages is significantly less accurate than using unique landing pages.

Secure analytics (metering) methods

All the methods described above (and some other methods not mentioned here, like sampling) have the central problem of being vulnerable to manipulation (both inflation and deflation). This means these methods are imprecise and insecure (in any reasonable model of security). This issue has been addressed in a number of papers , but to-date the solutions suggested in these papers remain theoretic, possibly due to lack of interest from the engineering community, or because of financial gain the current situation provides to the owners of big websites.

Chapter 13

Web Colors

Web colors are colors used in designing web pages, and the methods for describing and specifying those colors. Hexadecimal color codes begin with a hash (#).

Authors of web pages have a variety of options available for specifying colors for elements of web documents. Colors may be specified as an RGB triplet in hexadecimal format (a *hex triplet*); they may also be specified according to their common English names in some cases. Often a color tool or other graphics software is used to generate color values.

The first versions of Mosaic and Netscape Navigator used the X11 color names as the basis for their color lists, as both started as X Window System applications.

Web colors have an unambiguous colorimetric definition, sRGB, which relates the chromaticities of a particular phosphor set, a given transfer curve, adaptive whitepoint, and viewing conditions. These have been chosen to be similar to many real-world monitors and viewing conditions, so that even without color management rendering is fairly close to the specified values. However, user agents vary in the fidelity with which they represent the specified colors. More advanced user agents use color management to provide better color fidelity; this is particularly important for Web-to-print applications.

Hex triplet

A **hex triplet** is a six-digit, three-byte hexadecimal number used in HTML, CSS, SVG, and other computing applications, to represent colors. The bytes represent the red, green and blue components of the color. One byte represents a number in the range 00 to FF (in hexadecimal notation), or 0 to 255 in decimal notation. This represents the least (0) to the most (255) intensity of each of the color components. The hex triplet is formed by concatenating three bytes in hexadecimal notation, in the following order:

Byte 1: red value (color type red)
Byte 2: green value (color type green)
Byte 3: blue value (color type blue)

For example, consider the color where the red/green/blue values are decimal numbers: red=36, green=104, blue=160 (a greyish-blue color). The decimal numbers 36, 104 and 160 are equivalent to the hexadecimal numbers 24, 68 and A0 respectively. The hex

triplet is obtained by concatenating the 6 hexadecimal digits together, 2468A0 in this example.

Note that if any one of the three color values is less than 16 (decimal) or 10 (hex), it must be represented with a leading zero so that the triplet always has exactly six digits. For example, the decimal triplet 4, 8, 16 would be represented by the hex digits 04, 08, 10, forming the hex triplet 040810.

The number of colors that can be represented by this system is

$$256 \times 256 \times 256 = 16,777,216$$

An abbreviated, three (hexadecimal) digit form is sometimes used. Expanding this form to the six-digit form is as simple as doubling each digit: 09C becomes 0099CC. This allows each color value to cover its full range from 00 to FF. The three-digit form is described in the CSS specification, not in HTML. As a result, the three digit form in an attribute other than "style" is not interpreted as a valid color in some browsers.

Converting RGB to hexadecimal

RGB values are usually given in the 0–255 range; if they are in the 0–1 range, the values are multiplied by 255 before conversion. This number divided by 16 (integer division; ignoring any remainder) gives us the first hexadecimal digit (between 0 and F, where the letters A to F represent the numbers 10 to 15). The remainder gives us the second hexadecimal digit. For instance the RGB value 201 divides into 12 groups of 16, thus the first digit is C. A remainder of 9 gives us the hexadecimal number C9. This process is repeated for each of the three color values.

Conversion between number bases is a common feature of calculators, including both hand-held models and the software calculators bundled with most modern operating systems. Web-based tools specifically for converting color values are also available.

HTML color names

The HTML 4.01 specification defines sixteen named colors, as follows (names are defined in this context to be case-insensitive):

CSS1 / HTML3–4 / VGA color names

Name	Hex triplet	Red	Green	Blue	Hue	Satur	Light	Satur	Value	alias; CGA Number — Name
White	#FFFFFF	100%	100%	100%	0°	0%	100%	0%	100%	15 — white

Silver	#C0C0C0	75%	75%	75%	0°	0%	75%	0%	75%	7 — light gray
Gray	#808080	50%	50%	50%	0°	0%	50%	0%	50%	8 — dark gray
Black	#000000	0%	0%	0%	0°	0%	0%	0%	0%	0 — black
Red	#FF0000	100%	0%	0%	0°	100%	50%	100%	100%	12 — high red
Maroon	#800000	50%	0%	0%	0°	100%	25%	100%	50%	4 — low red
Yellow	#FFFF00	100%	100%	0%	60°	100%	50%	100%	100%	14 — yellow
Olive	#808000	50%	50%	0%	60°	100%	25%	100%	50%	6 — brown
Lime	#00FF00	0%	100%	0%	120°	100%	50%	100%	100%	green; 10 — high green
Green	#008000	0%	50%	0%	120°	100%	25%	100%	50%	2 — low green
Aqua	#00FFFF	0%	100%	100%	180°	100%	50%	100%	100%	cyan; 11 — high cyan
Teal	#008080	0%	50%	50%	180°	100%	25%	100%	50%	3 — low cyan
Blue	#0000FF	0%	0%	100%	240°	100%	50%	100%	100%	9 — high blue
Navy	#000080	0%	0%	50%	240°	100%	25%	100%	50%	1 — low blue
Fuchsia	#FF00FF	100%	0%	100%	300°	100%	50%	100%	100%	magenta; 13 — high magenta
Purple	#800080	50%	0%	50%	300°	100%	25%	100%	50%	5 — low magenta

These 16 were labelled as sRGB and included in the HTML 3.0 specification, which noted they were "the standard 16 colors supported with the Windows VGA palette."

X11 color names

In addition, a number of colors are defined by web browsers. A particular browser may not recognize all of these colors, but as of 2005 all modern general-use browsers support the full list. Many of these colors are from the list of X11 color names distributed with

the X Window System. These colors were standardized by SVG 1.0, and are accepted by SVG Full user agents. They are not part of SVG Tiny.

The list of colors actually shipped with the X11 product varies between implementations, and clashes with certain of the HTML names such as green. Furthermore, X11 colors are defined as simple RGB (hence, no particular color space), rather than sRGB. This means that the list of colors found in X11 (e.g. in /usr/lib/X11/rgb.txt) should not directly be used to choose colors for the web.

The list of web "X11 colors" from the CSS3 specification, along with their hexadecimal and decimal equivalents, is shown below, compare the alphabetical lists in the W3C standards. Note that this includes the common synonyms: aqua (HTML4/CSS 1.0 standard name) and cyan (common sRGB name), magenta (common sRGB name) and fuchsia (HTML4/CSS 1.0 standard name), gray (HTML4/CSS 1.0 standard name) and grey.

HTML name	Hex code R	G	B	Decimal code R	G	B
Red colors						
IndianRed	CD	5C	5C205	92	92	
LightCoral	F0	80	80240	128	128	
Salmon	FA	80	72250	128	114	
DarkSalmon	E9	96	7A233	150	122	
LightSalmon	FF	A0	7A255	160	122	
Crimson	DC	14	3C220	20	60	
Red	FF	00	00255	0	0	
FireBrick	B2	22	22178	34	34	
DarkRed	8B	00	00139	0	0	
Pink colors						
Pink	FF	C0	CB255	192	203	
LightPink	FF	B6	C1255	182	193	
HotPink	FF	69	B4255	105	180	
DeepPink	FF	14	93255	20	147	
MediumVioletRed	C7	15	85199	21	133	
PaleVioletRed	DB	70	93219	112	147	
Orange colors						
LightSalmon	FF	A0	7A255	160	122	
Coral	FF	7F	50255	127	80	
Tomato	FF	63	47255	99	71	
OrangeRed	FF	45	00255	69	0	
DarkOrange	FF	8C	00255	140	0	
Orange	FF	A5	00255	165	0	
Yellow colors						
Gold	FF	D7	00255	215	0	
Yellow	FF	FF	00255	255	0	
LightYellow	FF	FF	E0255	255	224	
LemonChiffon	FF	FA	CD255	250	205	
LightGoldenrodYellow	FA	FA	D2250	250	210	
PapayaWhip	FF	EF	D5255	239	213	
Moccasin	FF	E4	B5255	228	181	
PeachPuff	FF	DA	B9255	218	185	
PaleGoldenrod	EE	E8	AA238	232	170	
Khaki	F0	E6	8C240	230	140	
DarkKhaki	BD	B7	6B189	183	107	
Purple colors						

Lavender	E6	E6	FA	230	230	250
Thistle	D8	BF	D8	216	191	216
Plum	DD	A0	DD	221	160	221
Violet	EE	82	EE	238	130	238
Orchid	DA	70	D6	218	112	214
Fuchsia	FF	00	FF	255	0	255
Magenta	FF	00	FF	255	0	255
MediumOrchid	BA	55	D3	186	85	211
MediumPurple	93	70	DB	147	112	219
Amethyst	99	66	CC	153	102	204
BlueViolet	8A	2B	E2	138	43	226
DarkViolet	94	00	D3	148	0	211
DarkOrchid	99	32	CC	153	50	204
DarkMagenta	8B	00	8B	139	0	139
Purple	80	00	80	128	0	128
Indigo	4B	00	82	75	0	130
SlateBlue	6A	5A	CD	106	90	205
DarkSlateBlue	48	3D	8B	72	61	139
MediumSlateBlue	7B	68	EE	123	104	238

HTML name	Hex code			Decimal code		
	R	G	B	R	G	B
Green colors						
GreenYellow	AD	FF	2F	173	255	47
Chartreuse	7F	FF	00	127	255	0
LawnGreen	7C	FC	00	124	252	0
Lime	00	FF	00	0	255	0
LimeGreen	32	CD	32	50	205	50
PaleGreen	98	FB	98	152	251	152
LightGreen	90	EE	90	144	238	144
MediumSpringGreen	00	FA	9A	0	250	154
SpringGreen	00	FF	7F	0	255	127
MediumSeaGreen	3C	B3	71	60	179	113
SeaGreen	2E	8B	57	46	139	87
ForestGreen	22	8B	22	34	139	34
Green	00	80	00	0	128	0
DarkGreen	00	64	00	0	100	0
YellowGreen	9A	CD	32	154	205	50
OliveDrab	6B	8E	23	107	142	35
Olive	80	80	00	128	128	0
DarkOliveGreen	55	6B	2F	85	107	47
MediumAquamarine	66	CD	AA	102	205	170
DarkSeaGreen	8F	BC	8F	143	188	143
LightSeaGreen	20	B2	AA	32	178	170
DarkCyan	00	8B	8B	0	139	139
Teal	00	80	80	0	128	128

Blue/Cyan colors						
Aqua	00	FF	FF	0	255	255
Cyan	00	FF	FF	0	255	255
LightCyan	E0	FF	FF	224	255	255
PaleTurquoise	AF	EE	EE	175	238	238
Aquamarine	7F	FF	D4	127	255	212
Turquoise	40	E0	D0	64	224	208
MediumTurquoise	48	D1	CC	72	209	204
DarkTurquoise	00	CE	D1	0	206	209
CadetBlue	5F	9E	A0	95	158	160
SteelBlue	46	82	B4	70	130	180
LightSteelBlue	B0	C4	DE	176	196	222
PowderBlue	B0	E0	E6	176	224	230
LightBlue	AD	D8	E6	173	216	230
SkyBlue	87	CE	EB	135	206	235
LightSkyBlue	87	CE	FA	135	206	250
DeepSkyBlue	00	BF	FF	0	191	255
DodgerBlue	1E	90	FF	30	144	255
CornflowerBlue	64	95	ED	100	149	237
MediumSlateBlue	7B	68	EE	123	104	238
RoyalBlue	41	69	E1	65	105	225
Blue	00	00	FF	0	0	255
MediumBlue	00	00	CD	0	0	205
DarkBlue	00	00	8B	0	0	139
Navy	00	00	80	0	0	128
MidnightBlue	19	19	70	25	25	112

HTML name	Hex code R	G	B	Decimal code R	G	B
Brown colors						
Cornsilk	FF	F8	DC	255	248	220
BlanchedAlmond	FF	EB	CD	255	235	205
Bisque	FF	E4	C4	255	228	196
NavajoWhite	FF	DE	AD	255	222	173
Wheat	F5	DE	B3	245	222	179
BurlyWood	DE	B8	87	222	184	135
Tan	D2	B4	8C	210	180	140
RosyBrown	BC	8F	8F	188	143	143
SandyBrown	F4	A4	60	244	164	96

Goldenrod	DA	A5	20218	165	32
DarkGoldenrod	B8	86	0B184	134	11
Peru	CD	85	3F205	133	63
Chocolate	D2	69	1E210	105	30
SaddleBrown	8B	45	13139	69	19
Sienna	A0	52	2D160	82	45
Brown	A5	2A	2A165	42	42
Maroon	80	00	00128	0	0
White colors					
White	FF	FF	FF255	255	255
Snow	FF	FA	FA255	250	250
Honeydew	F0	FF	F0240	255	240
MintCream	F5	FF	FA245	255	250
Azure	F0	FF	FF240	255	255
AliceBlue	F0	F8	FF240	248	255
GhostWhite	F8	F8	FF248	248	255
WhiteSmoke	F5	F5	F5245	245	245
Seashell	FF	F5	EE255	245	238
Beige	F5	F5	DC245	245	220
OldLace	FD	F5	E6253	245	230
FloralWhite	FF	FA	F0255	250	240
Ivory	FF	FF	F0255	255	240
AntiqueWhite	FA	EB	D7250	235	215
Linen	FA	F0	E6250	240	230
LavenderBlush	FF	F0	F5255	240	245
MistyRose	FF	E4	E1255	228	225
Gray colors					
Gainsboro	DC	DC	DC220	220	220
LightGrey	D3	D3	D3211	211	211
Silver	C0	C0	C0192	192	192
DarkGray	A9	A9	A9169	169	169
Gray	80	80	80128	128	128

DimGray	69	69	69	105	105	105
LightSlateGray	77	88	99	119	136	153
SlateGray	70	80	90	112	128	144
DarkSlateGray	2F	4F	4F	47	79	79
Black	00	00	00	0	0	0

Web-safe colors

At one time many computer displays were only capable of displaying 256 colors. These may be dictated by the hardware or changeable by a "color table". When a color is found (e.g., in an image) that is not one available, a different one has to be used. This can be either using the closest color (fast) or dithering (slow, looks better).

There were various attempts to make a "standard" color palette. A set of colors was needed that could be shown without dithering on 256-color displays; the number 216 was chosen partly because computer operating systems customarily reserved sixteen to twenty colors for their own use; it was also selected because it allows exactly six shades each of red, green, and blue ($6 \times 6 \times 6 = 216$).

The list of colors is often presented as if it has special properties that render them immune to dithering. In fact, on 256-color displays applications can set a palette of any selection of colors that they choose, dithering the rest. These colors were chosen specifically because they matched the palettes selected by the then leading browser applications. Fortunately, there were not radically different palettes in use in different popular browsers.

"Web-safe" colors had a flaw in that, on systems such as X11 where the palette is shared between applications, smaller color cubes (5×5×5 or 4×4×4) were often allocated by browsers—thus, the "web safe" colors would actually dither on such systems. Better results were obtained by providing an image with a larger range of colors and allowing the browser to quantize the color space if needed, rather than suffer the quality loss of a double quantization.

As of 2007, personal computers typically have at least 16-bit color and usually 24-bit (TrueColor). Even mobile devices have at least 16-bit color, driven by the inclusion of cameras on cellphones. The use of "web-safe" colors has fallen into practical disuse, but persisted in culture.

The "web-safe" colors do not all have standard names, but each can be specified by an RGB triplet: each component (red, green, and blue) takes one of the six values from the

following table (out of the 256 possible values available for each component in full 24-bit color).

6 shades of each color

key	hex	decimal
0	00	0
3	33	51
6	66	102
9	99	153
C or (12)	CC	204
F or (15)	FF	255

The following table shows all of the "web-safe" colors, underlining the *really-safe* colors. (One shortcoming of the web-safe palette is its poor selection of light background colors.) The intensities at the low end of the range, especially the two darkest, are often hard to distinguish.

Color table

In the table below, each color code listed is a short-hand for the RGB value; for example, code 609 is equivalent to RGB code 102-0-153 or HEX code #660099.

Web-Safe Colors					
000	300	600	900	C00	*F00*
003	303	603	903	C03	*F03*
006	306	606	906	C06	F06
009	309	609	909	C09	F09
00C	30C	60C	90C	C0C	F0C
00F	30F	60F	90F	C0F	*F0F*
030	330	630	930	C30	F30
033	333	633	933	C33	F33
036	336	636	936	C36	F36
039	339	639	939	C39	F39
03C	33C	63C	93C	C3C	F3C
03F	33F	63F	93F	C3F	F3F
060	360	660	960	C60	F60
063	363	663	963	C63	F63
066	366	666	966	C66	F66
069	369	669	969	C69	F69
06C	36C	66C	96C	C6C	F6C
06F	36F	66F	96F	C6F	F6F
090	390	690	990	C90	F90

093	393	693	993	C93	F93
096	396	696	996	C96	F96
099	399	699	999	C99	F99
09C	39C	69C	99C	C9C	F9C
09F	39F	69F	99F	C9F	F9F
0C0	3C0	6C0	9C0	CC0	FC0
0C3	3C3	6C3	9C3	CC3	FC3
0C6	3C6	6C6	9C6	CC6	FC6
0C9	3C9	6C9	9C9	CC9	FC9
0CC	3CC	6CC	9CC	CCC	FCC
0CF	3CF	6CF	9CF	CCF	FCF
0F0	3F0	*6F0*	9F0	CF0	*FF0*
0F3	*3F3*	*6F3*	9F3	CF3	*FF3*
0F6	*3F6*	6F6	9F6	*CF6*	*FF6*
0F9	3F9	6F9	9F9	CF9	FF9
0FC	*3FC*	6FC	9FC	CFC	FFC
0FF	*3FF*	*6FF*	9FF	CFF	*FFF*

Safest web colors

Designers were often encouraged to stick to these 216 "web-safe" colors in their websites; however, 8-bit color displays were much more common when the 216-color palette was developed than they are now. David Lehn and Hadley Stern have since discovered that only 22 of the 216 colors in the web-safe palette are reliably displayed without inconsistent remapping on 16-bit computer displays. They called these 22 colors the "really safe" palette; it consists mainly of shades of green and yellow, as can be seen in the table above, where the "really safe" colors are underlined.

CSS colors

The Cascading Style Sheets language defines the same number of named colors as the HTML 4 spec, namely the 16 listed previously. Additionally, CSS 2.1 adds the 'orange' color name to the list:

Colors added in CSS 2.1

Name	Hex triplet	Red	Green	Blue	Hue	Satur	Light	Satur	Value	Alias
orange	#FFA500	100%	65%	0%	39°	100%	50%	—%	—%	

CSS 2, SVG and CSS 2.1 also allow web authors to use so-called *system colors*, which are color names whose values are taken from the operating system. This enables web authors to style their content in line with the operating system of the user agent. As of early 2004, it appears that the CSS3 color module will once again drop these values, marking them deprecated, but this may change.

The developing CSS3 specification will also introduce HSL color space values to style sheets:

```
/* RGB model */
p { color: #F00 } /* #rgb */
p { color: #FF0000 } /* #rrggbb */
p { color: rgb(255, 0, 0) } /* integer range 0 - 255 */
p { color: rgb(100%, 0%, 0%) } /* float range 0.0% - 100.0% */
/* RGB with alpha channel, added to CSS3 */
p { color: rgba(255, 0, 0, 0.5) } /* 0.5 opacity, semi-transparent */
/* HSL model, added to CSS3 */
p { color: hsl(0, 100%, 50%) } /* red */
p { color: hsl(120, 100%, 50%) } /* green */
p { color: hsl(120, 100%, 25%) } /* dark green */
p { color: hsl(120, 100%, 75%) } /* light green */
p { color: hsl(120, 50%, 50%) } /* pastel green */
/* HSL model with alpha channel */
p { color: hsla(120, 100%, 50%, 1) } /* green */
p { color: hsla(120, 100%, 50%, 0.5) } /* semi-transparent green */
p { color: hsla(120, 100%, 50%, 0.1) } /* very transparent green */
```

Accessibility

Some browsers and devices do not support colors. For these and blind and colorblind users, Web content depending on colors can be unusable or difficult to use. Both foreground and background color should be modified to avoid **black on black** effects. Similarly, most browsers show links as shades of blue by default; therefore, dark background colors, such as blue or navy, do not display well for such links.

Chapter 14

Comet (Programming)

Comet is a web application model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it. *Comet* is an umbrella term, encompassing multiple techniques for achieving this interaction. All these methods rely on features included by default in browsers, such as JavaScript, rather than on non-default plugins. The Comet approach differs from the original model of the web, in which a browser requests a complete web page at a time.

The use of Comet techniques in web development predates the use of the word *Comet* as a neologism for the collective techniques. Comet is known by several other names, including *Ajax Push*, *Reverse Ajax*, *Two-way-web*, *HTTP Streaming*, and *HTTP server push* among others.

Implementations

Comet applications attempt to eliminate the limitations of the page-by-page web model and traditional polling by offering real-time interaction, using a persistent or long-lasting HTTP connection between the server and the client. Since browsers and proxies are not designed with server events in mind, several techniques to achieve this have been developed, each with different benefits and drawbacks. The biggest hurdle is the HTTP 1.1 specification, which states that a browser should not have more than two simultaneous connections with a web server. However, holding one connection open for real-time events has a negative impact on browser usability: the browser may be blocked from sending a new request while waiting for the results of a previous request, e.g., a series of images. This can be worked around by creating a distinct hostname for real-time information, which is an alias for the same physical server.

Specific methods of implementing Comet fall into two major categories: streaming and long polling.

Streaming

An application using streaming Comet opens a single persistent connection from the client browser to the server for all Comet events. These events are incrementally handled and interpreted on the client side every time the server sends a new event, with neither side closing the connection.

Specific techniques for accomplishing streaming Comet include the following:

Hidden iframe

A basic technique for dynamic web application is to use a hidden iframe HTML element (an *inline frame*, which allows a website to embed one HTML document inside another). This invisible iframe is sent as a chunked block, which implicitly declares it as infinitely long (sometimes called "forever frame"). As events occur, the iframe is gradually filled with `script` tags, containing JavaScript to be executed in the browser. Because browsers render HTML pages incrementally, each `script` tag is executed as it is received.

One benefit of the iframe method is that it works in every common browser. Two downsides of this technique are the lack of a reliable error handling method, and the impossibility of tracking the state of the request calling process.

XMLHttpRequest

The XMLHttpRequest (XHR) object, the main tool used by Ajax applications for browser–server communication, can also be pressed into service for server–browser Comet messaging, in a few different ways.

In 1995, Netscape Navigator added a feature called "server push", which allowed servers to send new versions of an image or HTML page to that browser, as part of a multipart HTTP response, using the content type `multipart/x-mixed-replace`. Since 2004, Gecko-based browsers such as Firefox accept multipart responses to XHR, which can therefore be used as a streaming Comet transport. On the server side, each message is encoded as a separate portion of the multipart response, and on the client, the callback function provided to the XHR `onreadystatechange` function will be called as each message arrives. This functionality is only included in Gecko-based browsers, though there is discussion of adding it to Webkit.

Instead of creating a multipart response, and depending on the browser to transparently parse each event, it is also possible to generate a custom data format for an XHR response, and parse out each event using browser-side JavaScript, relying only on the browser firing the `onreadystatechange` callback each time it receives new data.

Ajax with long polling

None of the above streaming transports works across all modern browsers without negative side-effects in any. This forces Comet developers to implement several complex streaming transports, switching between them depending on the browser. Consequently many Comet applications use long polling, which is easier to implement on the browser side, and works, at minimum, in every browser that supports XHR. As the name suggests, long polling requires the client to poll the server for an event (or set of events). The browser makes an Ajax-style request to the server, which is kept open until the server has new data to send to the browser, which is sent to the browser in a complete response. The browser initiates a new long polling request in order to obtain subsequent events.

Specific technologies for accomplishing long-polling include the following:

XMLHttpRequest long polling

For the most part, XMLHttpRequest long polling works like any standard use of XHR. The browser makes an asynchronous request of the server, which may wait for data to be available before responding. The response can contain encoded data (typically XML or JSON) or Javascript to be executed by the client. At the end of the processing of the response, the browser creates and sends another XHR, to await the next event. Thus the browser always keeps a request outstanding with the server, to be answered as each event occurs.

Script tag long polling

While any Comet transport can be made to work across subdomains, none of the above transports can be used across different second-level domains (SLDs), due to browser security policies designed to prevent cross-site scripting attacks. That is, if the main web page is served from one SLD, and the Comet server is located at another SLD, Comet events cannot be used to modify the HTML and DOM of the main page, using those transports. This problem can be side-stepped by creating a proxy server in front of one or both sources, making them appear to originate from the same domain. However, this is often undesirable for complexity or performance reasons.

Unlike iframes or XMLHttpRequest objects, `script` tags can be pointed at any URI, and JavaScript code in the response will be executed in the current HTML document. This creates a potential security risk for both servers involved, though the risk to the data provider (in our case, the Comet server) can be avoided using JSONP.

A long-polling Comet transport can be created by dynamically creating `script` elements, and setting their source to the location of the Comet server, which then sends back JavaScript (or JSONP) with some event as its payload. Each time the script request is completed, the browser opens a new one, just as in the XHR long polling case. This method has the advantage of being cross-browser while still allowing cross-domain implementations.

History

Early Java applets

The ability to embed Java applets into browsers (starting with Netscape 2.0 in March 1996) made real-time communications possible, using a raw TCP socket to communicate between the browser and the server. This socket can remain open as long as the browser is at the document hosting the applet. Event notifications can be sent in any format — text or binary — and decoded by the applet.

First Comet applications

In March 2006, software engineer Alex Russell coined the term Comet in a post on his personal blog. The new term was a play on Ajax (Ajax and Comet both being household cleaners, common in the USA).

In 2006, while not really using the term Comet, some applications exposed those techniques to a wider audience: Meebo's multi-protocol web-based chat application enables users to connect to AOL, Yahoo, and Microsoft chat platforms through the browser; Google added web-based chat to Gmail; JotSpot, a startup since acquired by Google, built Comet-based real-time collaborative document editing. New Comet companies and enterprise solutions were created, such as the Java-based ICEfaces JSF framework (although they prefer the term "*Ajax Push*"). Others that had previously used Java-applet based transports switched instead to pure-JavaScript implementations.

Alternatives

Browser-native technologies are inherent in the term Comet. Attempts to improve non-polling HTTP communication have come from multiple sides:

- The HTML 5 draft specification produced by the Web Hypertext Application Technology Working Group (WHATWG) specifies so called server-sent events, it offers a new HTML element, `event-source` and a new data format called DOM event stream. Experimental implementation of this feature was introduced in Opera 9.
- The HTML 5 WebSocket API working draft specifies a method for creating a persistent connection with a server and receiving messages via an `onmessage` callback.
- The Bayeux protocol by the Dojo Foundation. It leaves browser-specific transports in place, and defines a higher-level protocol for communication between browser and server, with the aim of allowing re-use of client-side JavaScript code with multiple Comet servers, and allowing the same Comet server to communicate with multiple client-side JavaScript implementations. Bayeux is based on a publish/subscribe model, so servers supporting Bayeux have publish/subscribe built-in.
- The BOSH protocol by the XMPP standards foundation. It emulates a bidirectional stream between browser and server by using two synchronous HTTP connections.
- The JSONRequest object, proposed by Douglas Crockford, would be an alternative to the XHR object.
- Use of plugins, such as Java applets or the proprietary Adobe Flash (Java BlazeDS is a server plugin which streams events to Flash applications). These have the advantage of working identically across all browsers with the appropriate plugin installed and need not rely on HTTP connections, but the disadvantage of requiring the plugin to be installed
- Google announced a new Channel API for Google App Engine, implementing a Comet-like API with the help of a client javascript library on the browser. But it could be later replaced by HTML5 websockets.

Chapter 15

Mashup (Web Application Hybrid)

In Web development, a **mashup** is a Web page or application that uses and combines data, presentation or functionality from two or more sources to create new services. The term implies easy, fast integration, frequently using open APIs and data sources to produce enriched results that were not necessarily the original reason for producing the raw source data.

The main characteristics of the mashup are combination, visualization, and aggregation. It is important to make existing data more useful, moreover for personal and professional use.

To be able to permanently access the data of other services, mashups are generally client applications or hosted online. Since 2010, two major mashup vendors have added support for hosted deployment based on Cloud computing solutions; that are Internet-based computing, whereby shared resources, software, and information are provided to computers and other devices on demand, like the electricity grid.

In the past years, more and more Web applications have published APIs that enable software developers to easily integrate data and functions instead of building them by themselves. Mashups can be considered to have an active role in the evolution of social software and Web 2.0. Mashup composition tools are usually simple enough to be used by end-users. They generally do not require programming skills and rather support visual wiring of GUI widgets, services and components together. Therefore, these tools contribute to a new vision of the Web, where users are able to contribute.

The term *mashup* is also used to describe a *remix* of digital data.

History

The history of mashup can be backtracked by first understanding the broader context of the history of the Web. For Web 1.0 business model companies stored consumer data on portals and updated them regularly. They controlled all the consumer data, and the consumer had to use their products and services to get the information.

With the advent of Web 2.0 a new proposition was created, using Web standards that were commonly and widely adopted across traditional competitors and unlocked the consumer data. At the same time, mashups emerged allowing mixing and matching

competitor's API to create new services. The term isn't formally defined by any standard-setting body.

The first mashups used mapping services or photo services to combine these services with data of any kind and therefore create visualizations of the data. In the beginning, most mashups were consumer-based, but recently the mashup is to be seen as an interesting concept useful also to enterprises. Business mashups can combine existing internal data with external services to create new views on the data.

Mashups are in the ascending. As a statistic from Programmable Web found out in 2009 that three new mashups have been registered every single day for the last two years.

Types of mashups

There are many types of mashups, such as business mashups, consumer mashups, and data mashups. The most common type of mashup is the consumer mashup, aimed at the general public.

- *Business* (or *enterprise*) *mashups* generally define applications that combine their own resources, application and data, with other external Web services. They focus data into a single presentation and allow for collaborative action among businesses and developers. This works well for an agile development project, which requires collaboration between the developers and customer (or customer proxy, typically a product manager) for defining and implementing the business requirements. Enterprise mashups are secure, visually rich Web applications that expose actionable information from diverse internal and external information sources.

- *Consumer mashups* combines different data types. It combines data from multiple public sources in the browser and organizes it through a simple browser user interface.

- *Data mashups*, opposite to the consumer mashups, combine similar types of media and information from multiple sources into a single representation. The combination of all these resources create a new and distinct Web service that was not originally provided by either source.

By API Type

Mashups can also be categorized by the basic API type they use but any of these can be combined with each other or embedded into other applications.

Data types

- Indexed data (documents, weblogs, images, videos, shopping articles, jobs ...) used by Metasearch engines
- Cartographic and geographic data: Geolocation software, Geovisualization
- Feeds, podcasts: News aggregators

Functions

- Data converters: Language translators, Speech processing, URL shorteners...
- Communication: E-mail, Instant messaging, notification...
- Visual data rendering: Information visualization, diagrams
- Security related: electronic payment systems, ID identification...
- Editors

Mashups versus portals

Mashups and portals are both content aggregation technologies. Portals are an older technology designed as an extension to traditional dynamic Web applications, in which the process of converting data content into marked-up Web pages is split into two phases: generation of markup "fragments" and aggregation of the fragments into pages. Each markup fragment is generated by a "portlet", and the portal combines them into a single Web page. Portlets may be hosted locally on the portal server or remotely on a separate server.

Portal technology defines a complete event model covering reads and updates. A request for an aggregate page on a portal is translated into individual read operations on all the portlets that form the page ("`render`" operations on local, JSR 168 portlets or "`getMarkup`" operations on remote, WSRP portlets). If a submit button is pressed on any portlet on a portal page, it is translated into an update operation on that portlet alone (`processAction` on a local portlet or `performBlockingInteraction` on a remote, WSRP portlet). The update is then immediately followed by a read on *all* portlets on the page.

Portal technology is about server-side, presentation-tier aggregation. It cannot be used to drive more robust forms of application integration such as two-phase commit.

Mashups differ from portals in the following respects:

	Portal	Mashup
Classification	Older technology, extension to traditional Web server model using well-defined approach	Using newer, loosely defined "Web 2.0" techniques

Philosophy/Approach	Approaches aggregation by splitting role of Web server into two phases: markup generation and aggregation of markup fragments	Uses APIs provided by different content sites to aggregate and reuse the content in another way
Content dependencies	Aggregates presentation-oriented markup fragments (HTML, WML, VoiceXML, etc.)	Can operate on pure XML content and also on presentation-oriented content (e.g., HTML)
Location dependencies	Traditionally, content aggregation takes place on the server	Content aggregation can take place either on the server or on the client
Aggregation style	"Salad bar" style: Aggregated content is presented 'side-by-side' without overlaps	"Melting Pot" style - Individual content may be combined in any manner, resulting in arbitrarily structured hybrid content
Event model	Read and update event models are defined through a specific portlet API	CRUD operations are based on REST architectural principles, but no formal API exists
Relevant standards	Portlet behavior is governed by standards JSR 168, JSR 286 and WSRP, although portal page layout and portal functionality are undefined and vendor-specific	Base standards are XML interchanged as REST or Web Services. RSS and Atom are commonly used. More specific mashup standards such as EMML are emerging.

The portal model has been around longer and has had greater investment and product research. Portal technology is therefore more standardized and mature. Over time, increasing maturity and standardization of mashup technology will likely make it more popular than portal technology because it is more closely associated with Web 2.0 and lately Service-oriented Architectures (SOA). New versions of portal products are expected to eventually add mashup support while still supporting legacy portlet applications. Mashup technologies, in contrast, are not expected to provide support for portal standards.

Business mashups

Mashup use is expanding in the business environment. Business mashups are useful for integrating business and data services, as business mashups technologies provide the ability to develop new integrated services quickly, to combine internal services with external or personalized information, and to make these services tangible to the business user through user-friendly Web browser interfaces.

Business mashups differ from consumer mashups in the level of integration with business computing environments, security and access control features, governance, and the sophistication of the programming tools (mashup editors) used. Another difference

between business mashups and consumer mashups is a growing trend of using Business mashups in commercial software as a service (SaaS) offering.

Many of the providers of Business Mashups technologies have added SOA features.

Architectural aspects of mashups

The architecture of a mashup is divided into three layers:

- Presentation / User interaction: this is the user interface of mashups. The technologies used are HTML/XHTML, CSS, Javascript, Asynchronous Javascript and Xml (Ajax).

- Web Services: the products functionality can be accessed using the API services. The technologies used are XMLHTTPRequest, XML-RPC, JSON-RPC, SOAP, REST.

- Data: Handling the data like sending, storing and receiving. The technologies used are XML, JSON, KML.

Architecturally, there are two styles of mashups: Web-based and server-based. Whereas Web-based mashups typically use the user's Web browser to combine and reformat the data, server-based mashups analyze and reformat the data on a remote server and transmit the data to the user's browser in its final form.

Mashups appear to be a variation of a Facade pattern. That is: a software engineering design pattern that provides a simplified interface to a larger body of code (in this case the code to aggregate the different feeds with different APIs).

Mashups can be used with software provided as a service (SaaS).

After several years of standards development, mainstream businesses are starting to adopt Service-oriented Architectures (SOA) to integrate disparate data by making them available as discrete Web services. Web services provide open, standardized protocols to provide a unified means of accessing information from a diverse set of platforms (operating systems, programming languages, applications). These Web services can be reused to provide completely new services and applications within and across organizations, providing business flexibility.

Chapter 16

Microformat

A **microformat** (sometimes abbreviated **µF**) is a web-based approach to semantic markup which seeks to re-use existing HTML/XHTML tags to convey metadata and other attributes in web pages and other contexts that support (X)HTML, such as RSS. This approach allows software to process information intended for end-users (such as contact information, geographic coordinates, calendar events, and the like) automatically.

Although the content of web pages is technically already capable of "automated processing", and has been since the inception of the web, such processing is difficult because the traditional markup tags used to display information on the web do not describe what the information means. Microformats can bridge this gap by attaching semantics, and thereby obviate other, more complicated, methods of automated processing, such as natural language processing or screen scraping. The use, adoption and processing of microformats enables data items to be indexed, searched for, saved or cross-referenced, so that information can be reused or combined.

As of 2010 microformats allow the encoding and extraction of events, contact information, social relationships and so on. More are being developed.

Background

Microformats emerged as part of a grassroots movement to make recognizable data items (such as events, contact details or geographical locations) capable of automated processing by software, as well as directly readable by end-users. Link-based microformats emerged first. These include vote links that express opinions of the linked page, which search engines can tally into instant polls.

As the microformats community grew, CommerceNet, a nonprofit organization that promotes electronic commerce on the Internet, helped sponsor and promote the technology and support the microformats community in various ways. CommerceNet also helped co-found the Microformats.org community site.

Neither CommerceNet nor Microformats.org operates as a standards body. The microformats community functions through an open mailing list, and Internet relay chat (IRC) channel. Most of the existing microformats were created at the Microformats.org and the associated mailing list, by a process of gathering examples of web publishing

behaviour, then codifying it. Some other microformats (such as rel=nofollow and unAPI) have been proposed, or developed, elsewhere.

Plain Old Semantic HTML (POSH)

The phrase "plain old semantic HTML" has been found online as early as 1998, but the coinage of the acronym **POSH** used in connection with microformats occurred in April 2007 on the microformats irc channel. Semantic HTML focuses on the use of tags and attributes for semantic rather than presentational purposes. Its proponents discourage the use of tables for layout and the use of the `<b>` or `<br />` tags since these tags are purely presentational.

Technical overview

XHTML and HTML standards allow for the embedding and encoding of semantics within the attributes of markup tags. Microformats take advantage of these standards by indicating the presence of metadata using the following attributes:

- `class`
- `rel`
- `rev` (in one case, otherwise deprecated in microformats)

For example, in the text "The birds roosted at 52.48, -1.89" is a pair of numbers which may be understood, from their context, to be a set of geographic coordinates. With wrapping in spans (or other HTML elements) with specific class names (in this case `geo`, `latitude` and `longitude`, all part of the geo microformat specification):

```
The birds roosted at
   <span class="geo">
     <span class="latitude">52.48</span>,
     <span class="longitude">-1.89</span>
   </span>
```

software agents can recognize exactly what each value represents and can then perform a variety of tasks such as indexing, locating it on a map and exporting it to a GPS device.

In-context examples

Specific microformats

Several microformats have been developed to enable semantic markup of particular types of information.

- hAtom – for marking up Atom feeds from within standard HTML
- hCalendar – for events
- hCard – for contact information; includes:
    - adr – for postal addresses

    - geo – for geographical coordinates (latitude, longitude)

- hMedia - for audio/video content
- hNews - for news content
- hProduct – for products
- hRecipe - for recipes and foodstuffs.
- hResume – for resumes or CVs
- hReview – for reviews
- rel-directory – for distributed directory creation and inclusion
- rel-enclosure – for multimedia attachments to web pages
- rel-license – specification of copyright license
- rel-nofollow, an attempt to discourage third-party content spam (e.g. spam in blogs)
- rel-tag – for decentralized tagging (Folksonomy)
- xFolk – for tagged links
- XHTML Friends Network (XFN) – for social relationships
- XOXO – for lists and outlines

Microformats under development

Among the many proposed microformats, the following are undergoing active development:

- hAudio – for audio files and references to released recordings
- citation – for citing references
- currency – for amounts of money
- figure – for associating captions with images
- geo extensions – for places on Mars, the Moon, and other such bodies; for altitude; and for collections of waypoints marking routes or boundaries
- measure – For physical quantities, structured data-values

Uses of microformats

Using microformats within HTML code provides additional formatting and semantic data that applications can use. For example, applications such as web crawlers can collect data about on-line resources, or desktop applications such as e-mail clients or scheduling software can compile details. The use of microformats can also facilitate "mash ups" such as exporting all of the geographical locations on a web page into (for example) Google Maps to visualize them spatially.

Several browser extensions, such as Operator for Firefox and Oomph for Internet Explorer, provide the ability to detect microformats within an HTML document. When hCard or hCalendar are involved, such browser extensions allow to export them into formats compatible with contact management and calendar utilities, such as Microsoft Outlook. When dealing with geographical coordinates, they allow to send the location to maps applications such as Google Maps. Yahoo! Query Language can be used to extract microformats from web pages. On 12 May 2009, Google announced that they would be parsing the hCard, hReview and hProduct microformats, and using them to populate

search result pages. They have since extended this to use hCalendar for events and hRecipe for cookery recipes.

Microsoft expressed a desire to incorporate Microformats into upcoming projects; as have other software companies.

Alex Faaborg summarizes the arguments for putting the responsibility for microformat user interfaces in the web browser rather than making more complicated HTML:

- Only the web browser knows what applications are accessible to the user and what the user's preferences are
- It lowers the barrier to entry for web site developers if they only need to do the markup and not handle "appearance" or "action" issues
- Retains backwards compatibility with web browsers that don't support microformats
- The web browser presents a single point of entry from the web to the user's computer, which simplifies security issues

Evaluation of microformats

Various commentators have offered review and discussion on the design principles and practical aspects of microformats. Additionally, microformats have been compared to other approaches that seek to serve the same or similar purpose. From time to time, there is criticism of a single, or all, microformats. Documented efforts to advocate both the spread and use of microformats are known to exist as well. Opera Software CTO and CSS creator Håkon Wium Lie said in 2005 "We will also see a bunch of microformats being developed, and that's how the semantic web will be built, I believe." However, as of August 2008, Toby Inkster, author of the "Swignition" (formerly "Cognition") microformat parsing service pointed out that no new microformat specifications had been published for over three years.

Design principles

Computer scientist and entrepreneur, Rohit Khare stated that *reduce, reuse, and recycle* is "shorthand for several design principles" that motivated the development and practices behind microformats.[:71-72] These aspects can be summarized as follows:

- Reduce: favor the simplest solutions and focus attention on specific problems;
- Reuse: work from experience and favor examples of current practice;
- Recycle: encourage modularity and the ability to embed, valid XHTML can be reused in blog posts, RSS feeds, and anywhere else you can access the web.

Accessibility

Because some microformats make use of title attribute of HTML's `abbr` element to conceal machine-readable data (particularly date-times and geographical coordinates) in the "abbr design pattern", the plain text content of the element is inaccessible to those

screen readers that expand abbreviations. In June 2008, the BBC announced that it would be dropping use of microformats using the `abbr` design pattern because of accessibility concerns.

Comparison with alternative approaches

Microformats are not the only solution for providing "more intelligent data" on the web. Alternative approaches exist and are under development as well. For example, the use of XML markup and standards of the Semantic Web are cited as alternative approaches. Some contrast these with microformats in that they do not necessarily coincide with the design principles of "reduce, reuse, and recycle", at least not to the same extent.

One advocate of microformats, Tantek Çelik, characterized a problem with alternative approaches:

> “ Here's a new language we want you to learn, and now you need to output these additional files on your server. It's a hassle. (Microformats) lower the barrier to entry. ”

For some applications the use of other approaches may be valid. If one wishes to use microformat-style embedding but the type of data one wishes to embed does not map to an existing microformat, one can use RDFa to embed arbitrary vocabularies into HTML, for example: embedding domain-specific scientific data on the Web like zoological or chemical data where no microformat for such data exists. Furthermore, standards such as W3C's GRDDL allow microformats to be converted into data compatible with the Semantic Web.

Another advocate of microformats, Ryan King, put the compatibility of microformats with other approaches this way:

> “ Microformats provide an easy way for many people to contribute semantic data to the web. With GRDDL all of that data is made available for RDF Semantic Web tools. Microformats and GRDDL can work together to build a better web.

Chapter 17

Push Technology and Style Sheet (Web Development)

Push technology

Push technology, **server push**, describes a style of Internet-based communication where the request for a given transaction is initiated by the publisher or central server. It is contrasted with pull technology, where the request for the transmission of information is initiated by the receiver or client.

General use

Push services are often based on information preferences expressed in advance. This is called a publish/subscribe model. A client might "subscribe" to various information "channels". Whenever new content is available on one of those channels, the server would push that information out to the user.

Synchronous conferencing and instant messaging are typical examples of push services. Chat messages and sometimes files are pushed to the user as soon as they are received by the messaging service. Both decentralised peer-to-peer programs (such as WASTE) and centralised programs (such as IRC or XMPP) allow pushing files, which means the sender initiates the data transfer rather than the recipient.

Email is also a push system: the SMTP protocol on which it is based is a push protocol. However, the last step —from mail server to desktop computer— typically uses a pull protocol like POP3 or IMAP. Modern e-mail clients make this step seem instantaneous by repeatedly polling the mail server, frequently checking it for new mail. The IMAP protocol includes the IDLE command, which allows the server to tell the client when new messages arrive. The original BlackBerry was the first popular example of push technology for email in a wireless context.

Another popular type of Internet push technology was PointCast Network, which gained popularity in the 1990s. It delivered news and stock market data. Both Netscape and Microsoft integrated it into their software at the height of the browser wars, but it later faded away and was replaced in the 2000s with RSS (a pull technology).

Other uses are push enabled web applications including market data distribution (stock tickers), online chat/messaging systems (webchat), auctions, online betting and gaming, sport results, monitoring consoles and sensor network monitoring.

Technologies

HTTP server push

HTTP server push (also known as HTTP streaming) is a mechanism for sending data from a web server to a web browser. HTTP server push can be achieved through several mechanisms.

Generally the web server does not terminate a connection after response data has been served to a client. The web server leaves the connection open such that if an event is received, it can immediately be sent to one or multiple clients. Otherwise the data would have to be queued until the client's next request is received. Most web servers offer this functionality via CGI (e.g. Non-Parsed Headers scripts on Apache).

Another mechanism is related to a special MIME type called `multipart/x-mixed-replace`, which was introduced by Netscape in 1995. Web browsers would interpret this as a document changing whenever the server felt like pushing a new version to the client. It is still supported by Firefox, Opera and Safari today, but ignored by Internet Explorer. It can be applied to HTML documents, but also for streaming images in webcam applications.

The WHATWG Web Applications 1.0 proposal included a mechanism to push content to the client. On September 1, 2006, the Opera web browser implemented this new experimental technology in a feature called "Server-Sent Events." It is now being standardized as part of HTML5. Another related part of HTML5 is the WebSockets API, which allows a web server and client to communicate over a full-duplex TCP connection.

Pushlet

In this technique, the server takes advantage of persistent HTTP connections and leaves the response perpetually "open" (i.e. it never terminates the response), effectively fooling the browser into continuing in "loading" mode after the initial page load would normally be complete. The server then periodically sends snippets of javascript to update the content of the page, thereby achieving push capability. By using this technique the client doesn't need Java applets or other plug-ins to keep an open connection to the server. The clients will be automatically notified by new events, pushed by the server. One serious drawback to this method, however, is the lack of control the server has over the browser timing out. A page refresh is always necessary if a timeout occurs on the browser end.

Long polling

Long polling is a variation of the traditional polling technique and allows emulation of an information push from a server to a client. With long polling, the client requests

information from the server in a similar way to a normal poll. However, if the server does not have any information available for the client, instead of sending an empty response, the server holds the request and waits for some information to be available. Once the information becomes available (or after a suitable timeout), a complete response is sent to the client. The client will normally then immediately re-request information from the server, so that the server will almost always have an available waiting request that it can use to deliver data in response to an event. In a web/AJAX context, long polling is also known as Comet programming.

Long polling is itself not a push technology, but can be used under circumstances where a real push is not possible.

Flash XMLSocket relays

This technique, used by Cbox and other chat applications, makes use of the XMLSocket object in a single-pixel Adobe Flash movie. Under the control of JavaScript, the client establishes a TCP connection to a unidirectional relay on the server. The relay server does not read anything from this socket; instead it immediately sends the client a unique identifier. Next, the client makes an HTTP request to the web server, including with it this identifier. The web application can then push messages addressed to the client to a local interface of the relay server, which relays them over the Flash socket. The advantage of this approach is that it appreciates the natural read-write asymmetry that is typical of many web applications, including chat, and as a consequence it offers high efficiency. Since it does not accept data on outgoing sockets, the relay server does not need to poll outgoing TCP connections *at all*, making it possible to hold open tens of thousands of concurrent connections. In this model, the limit to scale is the TCP stack of the underlying server operating system.

Other technologies

The term Comet has been used to describe push technologies applied to Ajax web applications. It is used as an umbrella term for a combination of web technologies such as HTTP server push and long polling (see above).

XMPP is often used for push applications as well, especially the PubSub extensions. Apple uses this technology for its Mobile Me push support.

BOSH is a long-lived HTTP technique used in XMPP, but that can be used on the web. The specification (XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH)) reads: *This specification defines a transport protocol that emulates the semantics of a long-lived, bidirectional TCP connection between two entities (such as a client and a server) by efficiently using multiple synchronous HTTP request/response pairs without requiring the use of frequent polling or chunked responses.*

Style sheet (web development)

Web **style sheets** are a form of separation of presentation and content for web design in which the markup (i.e., HTML or XHTML) of a webpage contains the page's semantic content and structure, but does not define its visual layout (style). Instead, the style is defined in an external stylesheet file using a style sheet language such as CSS or XSL. This design approach is identified as a "separation" because it largely supersedes the antecedent methodology in which a page's markup defined both style and structure.

The philosophy underlying this methodology is a specific case of separation of concerns.

Benefits

Separation of style and content has many benefits, but has only become practical in recent years due to improvements in popular web browsers' CSS implementations.

Speed

Overall, users experience of a site utilising style sheets will generally be quicker than sites that don't use the technology. 'Overall' as the first page will probably load more slowly – because the style sheet AND the content will need to be transferred. Subsequent pages will load faster because no style information will need to be downloaded – the CSS file will already be in the browser's cache.

Maintainability

Holding all the presentation styles in one file significantly reduces maintenance time and reduces the chance of human errors, thereby improving presentation consistency. For example, the font color associated with a type of text element may be specified — and therefore easily modified — throughout an entire website simply by changing one short string of characters in a single file. The alternate approach, using styles embedded in each individual page, would require a cumbersome, time consuming, and error-prone edit of every file.

Accessibility

Sites that use CSS with either XHTML or HTML are easier to tweak so that they appear extremely similar in different browsers (Internet Explorer, Mozilla Firefox, Opera, Safari, etc.).

Sites using CSS "degrade gracefully" in browsers unable to display graphical content, such as Lynx, or those so very old that they cannot use CSS. Browsers ignore CSS that they do not understand, such as CSS 3 statements. This enables a wide variety of user agents to be able to access the content of a site even if they cannot render the stylesheet or are not designed with graphical capability in mind. For example, a browser using a refreshable braille display for output could disregard layout information entirely, and the user would still have access to all page content.

Customization

If a page's layout information is all stored externally, a user can decide to disable the layout information entirely, leaving the site's bare content still in a readable form. Site authors may also offer multiple stylesheets, which can be used to completely change the appearance of the site without altering any of its content.

Most modern web browsers also allow the user to define their own stylesheet, which can include rules that override the author's layout rules. This allows users, for example, to bold every hyperlink on every page they visit.

Consistency

Because the semantic file contains only the meanings an author intends to convey, the styling of the various elements of the document's content is very consistent. For example, headings, emphasized text, lists and mathematical expressions all receive consistently applied style properties from the external stylesheet. Authors need not concern themselves with the style properties at the time of composition. These presentational details can be deferred until the moment of presentation.

Portability

The deferment of presentational details until the time of presentation means that a document can be easily re-purposed for an entirely different presentation medium with merely the application of a new stylesheet already prepared for the new medium and consistent with elemental or structural vocabulary of the semantic document. A carefully authored document for a web page can easily be printed to a hard-bound volume complete with headers and footers, page numbers and a generated table of contents simply by applying a new stylesheet.

Practical disadvantages today

Currently specifications (for example, XHTML, XSL, CSS) and software tools implementing these specification are only reaching the early stages of maturity. So there are some practical issues facing authors who seek to embrace this method of separating content and style.

Narrow adoption without the parsing and generation tools

While the style specifications are quite mature and still maturing, the software tools have been slow to adapt. Most of the major web development tools still embrace a mixed presentation-content model. So authors and designers looking for GUI based tools for their work find it difficult to follow the semantic web method. In addition to GUI tools, shared repositories for generalized stylesheets would probably aid adoption of these methods.

Chapter 18

Web Workers and WebSockets

Web Workers

Web Workers defines an API for running scripts, basically JavaScript, in the background independently of any user interface scripts. This allows for long-running scripts that are not interrupted by scripts that respond to clicks or other user interactions, and allows long tasks to be executed without yielding to keep the page responsive.

Web workers are relatively heavy-weight, and are not intended to be used in large numbers as they could hog system resources. Browser implementation of Web Workers are different.

Generally, web workers are expected to be long-lived, have a high start-up performance cost, and a high per-instance memory cost.

Overview

Web workers allows for concurrent execution of the browser threads and one or more JavaScript threads running in the background. The browser which follows a single thread of execution will have to wait on JavaScript programs to finish executing before proceeding and this may take significant time which the programmer may like to hide from the user. It allows for the browser to continue with normal operation while running in the background. Web worker specification is a separate specification from HTML5 specification but can be used with HTML5.

There are two types of web workers : **Dedicated worker** and **Shared worker**

When web workers run in the background, they do not have direct access to the DOM but communicate with the document by message passing. It allows for a multi-threaded execution of JavaScript programs.

Features

Web workers interact with the document via message passing. The following code loads a JavaScript file

```
var worker = new Worker("worker_script.js");
```

To send message to the worker, the `postMessage` method of the worker object is used as shown below

```
worker.postMessage("Hello World!");
```

The method `onmessage` is used to retrieve information from the worker

```
worker.onmessage = function(event) {
                alert("received message " + event.data);
                doSomething();
        }

        function doSomething() {
                //do work
                postMessage("Work done!");
        }

        worker.close
```

Once a worker is terminated, it goes out of scope and a new worker has to be created if needed.

Example

The simplest use of web workers is for performing a computationally expensive task without interrupting the user interface.

In this example, the main document spawns a web worker to compute prime numbers, and progressively displays the most recently found prime number.

The main page is as follows:

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>Worker example: One-core computation</title>
 </head>
 <body>
  <p>The highest prime number discovered so far is: <output
id="result"></output></p>
  <script type="text/javascript">
   var worker = new Worker('worker.js');
   worker.onmessage = function (event) {
     document.getElementById('result').textContent = event.data;
   };
  </script>
 </body>
</html>
```

The `Worker()` constructor call creates a web worker and returns a `Worker` object representing that web worker, which is used to communicate with the web worker. That object's `onmessage` event handler allows the code to receive messages from the web worker.

The web worker itself is as follows:

```
var n = 1;
search: while (true) {
  n += 1;
  for (var i = 2; i <= Math.sqrt(n); i += 1)
    if (n % i == 0)
     continue search;
  // found a prime!
  postMessage(n);
}
```

To send a message back to the page, the `postMessage()` method is used to post a message when a prime is found.

Support

If the browser supports web workers, a worker property will be available on the global window object . The window property will be undefined if the browser does not support it.

The following example code checks for web worker support on a browser

```
  function detect_web_worker() {
                return !!window.Worker;
        }
```

Third party libraries for JavaScript such as Modernizr can also be used to detect browser functionality for web workers. Web workers is currently supported by Safari, Chrome, Opera and Mozilla Firefox . Internet Explorer, iOS 4.2, and Android 2.3 do not support this.

WebSockets

WebSocket is a technology providing for bi-directional, full-duplex communications channels, over a single Transmission Control Protocol (TCP) socket. It is designed to be implemented in web browsers and web servers but it can be used by any client or server application. The WebSocket API is being standardized by the W3C and the WebSocket protocol is being standardized by the IETF. Since ordinary TCP connections to ports other than 80 are frequently blocked by administrators outside of home environments it can be used as a way to overcome these restrictions and provide similar functionality with

some additional protocol overhead while multiplexing several WebSocket services over a single TCP port.

For the client side, WebSocket was to be implemented in Firefox 4, Google Chrome 4, Opera 11 and Safari 5, as well as the mobile version of Safari in iOS 4.2. However, although present, support is now disabled by default in Firefox and Opera, due to concerns over security vulnerabilities.

WebSocket Protocol Handshake

To establish a WebSocket connection, the client sends a WebSocket handshake request, and the server sends a WebSocket handshake response, as shown in the following example:

Proxy traversal

WebSocket protocol client implementations try to detect if the user agent is configured to use a proxy when connecting to destination host and port and, if it is, uses HTTP CONNECT method to set up a persistent tunnel.

While the WebSocket protocol itself is unaware of proxy servers and firewalls, it features an HTTP-compatible handshake so that HTTP servers can share their default HTTP and HTTPS ports (80 and 443) with a WebSocket gateway or server. The WebSocket protocol defines a ws:// and wss:// prefix to indicate a WebSocket and a WebSocket Secure connection, respectively. Both schemes use an HTTP upgrade mechanism to upgrade to the WebSocket protocol. Some proxy servers are harmless and work fine with WebSocket; others will prevent WebSocket from working correctly, causing the connection to fail. In some cases additional proxy server configuration may be required, and certain proxy servers may need to be upgraded to support WebSocket.

If unencrypted WebSocket traffic flows through an explicit or a transparent proxy server on its way to the WebSocket server, then, whether or not the proxy server behaves as it should, the connection is almost certainly bound to fail today (as WebSocket become more mainstream, proxy servers may become WebSocket aware). Therefore, unencrypted WebSocket connections should be used only in the simplest topologies.

If an encrypted WebSocket connection is used, then the use of Transport Layer Security (TLS) in the WebSocket Secure connection ensures that an HTTP CONNECT command is issued when the browser is configured to use an explicit proxy server. This sets up a tunnel, which provides low-level end-to-end TCP communication through the HTTP proxy, between the WebSocket Secure client and the WebSocket server. In the case of transparent proxy servers, the browser is unaware of the proxy server, so no HTTP CONNECT is sent. However, since the wire traffic is encrypted, intermediate transparent proxy servers may simply allow the encrypted traffic through, so there is a much better chance that the WebSocket connection will succeed if WebSocket Secure is used. Using encryption is not free of resource cost, but often provides the highest success rate.

A recent update to the draft (version 76) broke compatibility with reverse-proxies and gateways because the 8 bytes of data the client must send after the headers is not advertised in a Content-Length header, so the intermediates won't forward that data until the handshake completes. And since the handshake needs those 8 bytes to complete, the handshake never completes and deadlocks. In current state of affairs, it's not advisable to modify such intermediate components to support this non-standard HTTP behaviour because doing so would render the components vulnerable to HTTP smuggling attacks, since an attacker would just have to pretend trying to upgrade to the WebSocket protocol in a request to be able to send more data than the target plain HTTP server can parse, possibly bypassing some mandatory security filtering. It is not known if this recent breakage will be worked around in a new draft or not.

URL scheme

The WebSocket protocol specification defines two new URI schemes, **ws:** and **wss:**, for unencrypted and encrypted connections. Apart from the scheme name, the rest of the URI components are defined to use URI generic syntax.

Browsers supporting WebSocket

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.

- Firefox 4
- Opera 11

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

Microsoft supports the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.

Implementation status

Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
draft-hixie-thewebsocketprotocol-75			4	5.0.0		
draft-hixie-thewebsocketprotocol-76 **draft-ietf-hybi-thewebsocketprotocol-00**		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
draft-ietf-hybi-thewebsocketprotocol-06	HTML5 Labs	dev				