Objective

• This course is to learn an integrated knowledge of Web design, analysis, implementation, and application.
• Topics such as Web software architecture, client and server technologies, Object Web, dynamic executable scheduling, web-based knowledge base, and Web Security will be covered.
• The recent research and results will be presented. New approaches and current issues will be discussed.

Introduction

• A course comprises the necessary knowledge to make decision on which Web technology is employed.
• This course is for those who want to learn how to do Web programming professionally.
• You may be one of these people:
  – a legacy programmer,
  – an armchair jockey,
  – a new programmer.

Overview of the Technology

• The word language is quickly becoming difficult to fit into the scope of what are fast becoming things called technologies.
• Electronic commerce is driving the expansion of the Internet.
Course Organization

- An overview of the Internet
- Web programming
  - HTML
  - Dynamic HTML
  - JavaScript
  - Visual Basic
  - PERL
  - The Basics of Java

Course Organization

- Web databases
- Security and E-Commerce
- XML
- Other topics

Tools You Will Need

- Internet browser and Internet access
- Space on a server
- Web page editor
- PERL, Java, Visual Basic,
  - www.wiley.com/compbooks/cintron

CS 898N – Advanced World Wide Web Technologies

Lecture 2: Overview of the Internet

Chin-Chih Chang
chang@cs.twsu.edu

The Internet Defined

- The Internet is a worldwide network of computers and data communications equipment which opens access to everyone.
- Originally put together as a communication network by U.S. university research facilities combined with U.S. Department of Defense funding, it was first known as the ARPANet.
- The Internet is now a worldwide enterprise that belongs to the public.
Growing of the Internet

- In 1988 when the Internet was first plugged into a T1 backbone, there were a total of about 50,000 hosts.
- In 1993, when the World Wide Web came online, the number of hosts had just passed 1 million mark.
- In 2001, there are more than 100 million hosts on the Internet. The Internet is growing in a fast speed.
- The most recent survey can be found at http://www.isc.org/ds/.

Who Pays for the Internet

What’s the Internet Made of?

- The Internet is based on a set of Network Access Points (NAPs) which any private backbone operator can hook into.
- At the highest level we find fastest “Optical carrier” backbone-level connections running hundreds of megabits per second.
- Speed standards varies from Transmission Level 1 (T1) - 1.544Mbps to Optical Carrier Level 768 (OC-768) – 39.8Gbps.

What’s the Internet Made of?

- The NAP network is really a symbolic network representing interconnectivity.
- Typically, an actual NAP-connected backbone provider is a telephone company (Figure 1.5).
- Below the level of NAP backbone provider we have the Regional Network Operators (Figure 1.6).
- These regional operators usually provide service to less densely populated areas.

What’s the Internet Made of?

- At the lowest level we have the local ISP. These companies usually have just one location.
- The slowest connections an ISP can offer range from the T-1 leased line at 1.544Mbps down through the 128kbps ISDN (Integrated Services Digital Network) or 56 kbps leased line.
What’s the Internet Made of?

- The evolution of modem speeds has been slowly but steadily increasing, as illustrated in Table 1.2.
- In 1997, 56K baud (bps) has become a standard dialup speed.
- In 1998, ASDL (Asymmetric Digital Subscriber Line) and cable modem are available for more than T1 speed.
- Cable modem offers fast download but slow upload.

What is an Intranet?

- An Internet is a network that lies between networks and unites them. An intranet is a private network that is contained within an enterprise.
- The main purpose of an intranet is to share company information and computing resources among employees.
- In an intranet, we create Web pages that provide company information and training.

What is an Intranet?

- This type of network usually allows users to access the Internet, but is blocked off from unauthorized access by a firewall.
- A firewall is a server set up so that all traffic into and out of the network has to go through it, and it set up to either permit or deny traffic according to where it’s coming from, where it’s going, and what it is.

What is a Computer Network?

- A computer network is a bunch of computers connected together.
- The Local Area Network (LAN) is usually confined to a floor or building (Figure 1.8).
- The Wide Area Network (WAN) is composed of two or more LANs (Figure 1.9).

Wide Area Network (WAN)

- A WAN could be distributed throughout a large building, several buildings, or even between cities or countries.
- A trend is for corporations to pay an ISP to connect them
together, which is much less expensive than leasing long-distance high-speed lines. This is called a Virtual Private Network (VPN).

• A network topology refers to the way in which a network is laid out.

Types of LANS

• Ethernet is a standard for communications and not a type of connector. Ethernet connectors can be coaxial cable, fiber optic, or twisted pair wire.
• The Ethernet has three speed standards: 10 Mbps, 100 Mbps and 1 Gbps
• Three major types of LAN topologies are:
  – Bus Network (Figure 1.10)
  – Hub (Star) Network (Figure 1.11)
  – Ring Network (Figure 1.12)

How do Networks Communicate with Each other?

• Connecting hubs to buses is done through a kind of equipment referred to as routers.
• Routers fall into a few separate classes of equipment called gateways, routers, and bridges.
• A gateway is a network point that acts as an entrance to another network.
• A computer server acting as a gateway node is often also acting as a proxy server and a firewall server.

How do Networks Communicate with Each other?

• A proxy server is a server that acts as an intermediary between a workstation user and the Internet so that the enterprise can ensure security, administrative control, and caching service.
• A bridge connects two similar LANs using same type of communications.
• A brouter is a bridge-routers.
For more information about the Internet, check www.isoc.org.

Client/Server

- A program running on the end-user workstation is called a **client**.
- A program running on the service part is called a **server**.
- Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request.

The Internet as a Virtual World

- The Internet is location independent. The locations are handled by the network, and is invisible to you.
- Search Engines (Table 1.3) are popular entrance points to the Internet. There is full of information on the Internet.
- **Spam** is a word that describes unwanted Internet content.
- Newsgroups and many other services are available on the Internet.

A Network of Networks

- The Internet has been made possible by the use of standard data communications protocols. Every computer on the Internet understands this specific set of protocols.
- A communication protocol is a standardized method for transmitting data between computers in a way that it can be sent, received, and processed without error.

World Wide Web

- The World Wide Web (WWW, Web) was originally designed by Tim Berners-Lee as a global hypertext project in 1990.
- Hypertext is a method of linking text together.
- Hypertext Markup Language (HTML) is a language for
• The purpose of the original Web browser, Mosaic, was to display formatted hypertext.

**World Wide Web**

• The theory is the hypertext can create a unified knowledge base that united all the information in the universe into an interlinked whole.

• If we were to cross-reference every relevant piece of information with every other, our Web documents would represent a complete and formidable knowledge engine.

**World Wide Web**

• When the Web documents can be further forms of data storage, such as audio and video, we come up with a larger concept called *hypermedia*, all kept in a world called *hyperspace*.

• HTTP is the primary protocol that all Web browsers are programmed to use.

• HTTP is HyperText Transfer Protocol. This is the protocol for transferring hypertext information on the Internet.

**Domain**

• A domain was one of the main hosts or subnetworks of the Internet, a domain name was a way to access that specific host or network.

• A domain name is the central part of the Internet address.

• Domain names are split into two parts: the first (or top) level and second level.

• The second-level domain is the name you choose.

**Domain Name**

• The first-level domain is the extension. The first-level domain is assigned according to what kind of domain it represents.

• You can check the up-to-the-minute status of all of the top-level domains at [www.iana.org/domain-names.html](http://www.iana.org/domain-names.html)

• You can register the domain name of your choice at Network Solutions, the registration arm of InterNIC (Internet Network
URL

- The Uniform Resource Locator (URL) is used to find an exact target within a domain.
- URL can be broken down into five parts:
  - the protocol designator such as http:// or ftp://,
  - the subdomain name,
  - the actual domain name,
  - the port number,
  - the path of a specific file to access.

The Internet and URL

- IP means Internet Protocol. Every domain on the Internet is assigned a unique number. This number is 12 digitals long (four sets of 3 digits each) and is called the IP address.
- When you type in a domain this is translated into the 12-digital number.
- The organizations maintaining the IP address list publish an Internet phone book.

From Browser to Server

- The browser calls a program to make a dial-up connection to your local ISP access number.
- The provider’s end runs a program that constantly checks for incoming calls for the connection.
- Routers use the numeric addresses to route traffic from source to destination and back.
- The server runs a program awaiting an incoming request.

Server

- A server is a computer with two features:
  - It's hardwired into the Internet,
  - It has a great deal of specialized server software.
- To set up a Web server, you need a server software.
- The Apache server is available to download without cost at www.apache.org
• You can have a series of options of services.

Internet Architecture
• The architecture is a specification that defines exactly how electronic communication will occur between computers on the Internet.
• The Internet architecture is based on the network architecture.
• The **OSI 7-Layer Reference Model** [ISO,1984] is a guide that specifies what each layer should do, but not how each layer is implemented.

Internet Architecture
• OSI 7-Layer Reference Model
  – Application layer: various applications (ftp, http)
  – Presentation layer: present data in a meaningful format
  – Session layer: provide session semantics (RPC)
  – Transport layer: reliable end-to-end byte stream (TCP)
  – Network layer: unreliable end-to-end transmission of packets

Internet Architecture
• OSI 7-Layer Reference Model (continued)
  – Data link layer: reliable transmission of frames
  – Physical layer: unreliable transmission of raw bits

• The conceptual intention here is that each the software which implements each layer communicates with its **Peer Layer** software, using services provided by the lower layers.

Layered Architecture
• TCP/IP stands for a combination of Transport Control Protocol/Internet Protocol.
• The TCP layer takes responsibility for ensuring the communication is completed.
• The TCP layer converts messages that are handed to it by the application layer into TCP format by adding the TCP control information to the front of message, now called a TCP header.

Layered Architecture
• The TCP layer then hands the whole message over the IP layer.
• The IP layer takes responsibility for ensuring that the
Communications are correctly routed.

• The physical layer performs the transmission of the data.
• At each level, the protocol handling a data block either adds its protocol-specific information or removes it from that data.

**Communications Protocols**

• The connection between browser and provider is accomplished in four steps:
  – The modem connection,
  – Login
  – IP connection
  – HTTP connection
• Refer to these sites for more information: [www.internic.net](http://www.internic.net), [www.iana.org](http://www.iana.org), [www.arin.net](http://www.arin.net), [www.nic.gov](http://www.nic.gov)

**The HTTP Connection**

• The HTTP protocol is text-based.
• HTTP headers:
  – GET: identifies the request as HTTP version 1.1.
  – Accept: identifies what image formats are accepted.
  – Accept-Language: specifies the language used.
  – Accept-Encoding: specifies the data compression.

**The HTTP Connection**

• HTTP headers (continued):
  – User-Agent: identifies the user agent.
  – Host: requests the homepage.
  – Connection: specifies to keep the connection open.
  – Extension: Something about security.

**The Domain Name Server**

• The provider’s end convert the domain name for the Web page requested into an IP address.
• The originating server calls a program called a *name resolver*. This program accesses a table on the server with the addresses of the local name server.
• The name server will either have the IP address on the requested DNS (Domain Name Server) or query a remote name server.
The Domain Name Server

- The domain name system is set up in a hierarchical fashion.
- The application eventually looks up for the root name server. The root name server will replay the address resolution request to the appropriate server of the requested domain.
- The scheme follows the network numbering scheme, also called *dotted decimal notation*.
- The **ping** command checks if a machine responds.

The Domain Name Server

- IP addresses are handed out according to the size of the network.
- The actual number handed out is called the *network* (or *subnet*) *mask*, because the network addresses will have that part as a fixed value with the rest of the address variable.

Packet Switching

- The Internet is a packet-switched network. All data is packaged in TCP and IP headers and sent through routers.
- A packet is a block of data packaged for transmission. Data packets are smaller pieces of a larger block of data that is broken down and sent in the individual packets, then received and reassembled.

Communication Cycle

- In packet switching, individual packets of data may go one way or another, their route switched according to what is most efficient at that time.
- In page 48 an example of the communication cycle is illustrated.

The Internet as a Managed Network

- There are two categories of organizations trying to keep the Internet in order:
  - The Internet Society (ISOC) consisting mostly of individual members.
  - The W3C (World Wide Web Consortium) consisting entirely of corporate memberships.
- ISOC is also at the top level of a hierarchy of Internet organizations.
Internet Organizations

- ISOC provides leadership in addressing issues that confront the future of the Internet, and is the home for the groups responsible for Internet infrastructure standards, including the Internet Engineering Task Force (IETF) and the Internet Architecture Board (IAB).
- The IAB is a technical advisory group of the Internet Society.

Internet Organizations

- The IETF is engaged in the development of new Internet standard specifications.
- The Internet Engineering Steering Group (IESG) is part of IETF and is responsible for technical management of IETF activities and the Internet standards process.
- The Internet Research Task Force (IRTF) is also a part of ISOC. Its purpose is a more farsighted version of IESG.
- The Internet Assigned Numbers Authority (IANA) is responsible for assigning a unique identifier to everything involving a standard or protocol that needs one.
- The World Wide Web Consortium (W3C) is to develop common protocols to enhance the interoperability and lead the evolution of the World Wide Web.
- RFC means Request for Comment. RFCs contain all of the protocols in use throughout the Internet.
• The IETF recommends and approves working groups that are run by the IESG under the IETF.
• These working groups tackle the task of putting together a specification:
  – Internet draft
  – Proposed standard
  – Draft standard
  – Internet Standard

CS 898N – Advanced World Wide Web Technologies
Lecture 4: Programming, Scripting, and Applets
Chin-Chih Chang
chang@cs.twsu.edu

Programming Languages
• You write your basic Web page in HTML. You find a Java applet that does some cool stuff. Maybe you need the user to fill out a form and you find a CGI script in the public domain. You want to validate the input, so you add a little JavaScript. You may want to try the latest technologies: cascading style sheet (CSS), Dynamic HTML, or XML.

Programming Basics
• Programming starts with the specification for a programming language. These specifications are usually much harder to read than the language itself and have to be deciphered by people who write books on programming.
• In today’s programming world, language diversity abounds. But all languages are limited to doing what the CPU instruction set is capable of.

Programming Basics
• In computer programming, mathematical proficiency is important.
• The machine language is the language that the machine can understand. The machine language instructions are actually broken down into microcode as they are executed.
• Microcode is based on the specific architecture of the CPU.
Compilers

• There are high-level languages and low-level languages, sometimes also referred to as first-, second-, third-, and fourth-generation languages.
• The closer the instructions in the language correspond with the CPU instruction set, the lower level is.
• The high-level language instruction could translate into hundreds, or thousands of low-level instructions.

Compilers

• A compiler is written, which translates the text-based code into machine language.
• When a compiler goes through a program, it separates data and instructions out.
• Some languages require you to declare all of your data names, variables, blocks, or sections.
• Many compilers are written in assembly language.

Compilers

• Assembly language is machine code made human readable, but it still reflects machine instructions.
• A compiler that compiles assembly language code into machine code is called an assembler.
• There are two stages to compiling a program, called passes.

Compilers

• The first pass compiles the human-written program into what we call pseudocode, or intermediate code, or object code.
• The compiler’s first pass translates every piece of data and instruction into tables and symbolic codes.
• The compiler performs its second pass to translate intermediate code into machine code.

Compilers

• In some cases, a runtime can do the second compiler pass and then execute the result.
• For example, Java’s bytecode is pseudocode run by Java Virtual Machine (Java VM) runtime system.
• See How does a Compiler Work on the page 71.
Object Code and Subroutines

- Subroutines are those routines which are called from the main program.
- Subroutines would be linked to the main program at the object code stage before being compiled to a machine language program.
- Linking means resolving data addresses between programs.

Object Code and Subroutines

- Subroutines have unresolved data references that have been resolved before program can be executed.
- Object code is code that defines these data and instruction sections as objects in a way that these references can be resolved, or linked together.
- In some languages the second-pass compilers are also called linkers.

Runtime Systems

- Rather than writing the code which do low-level system tasks for every program, programmers load these routines into memory at runtime. This is called a runtime system.
- Some languages use a runtime system to provide services to the running program.

Runtime Systems

- On UNIX it is called a daemon. On DOS it is a TSR (Terminate and Stay Resident). On a Novell Network it can be an NLM (Netware Loadable Module).
- The services a runtime provides are several. The simplest runtime may just handle system calls.
- A runtime may also execute pseudocode. Like in Java, once the code is compiled, it can be run on different platforms.

Scripting

- Scripting languages such as VBScript and JavaScript are all interpreted at runtime. They are compiled and executed on the spot by the browser.
• Scripting languages have grown out of traditional programming languages from the need to control command flow at the OS prompt level.

Scripting
• The early scripting languages were those controlling shell commands in UNIX.
• Server-side scripts are now coming into play in the form of applications like Active Server Pages.
• Active Server Pages is a scheme from Microsoft that means the server can does something active with the script on the page.

Scripting
• Scripting languages trade speed of execution for flexibility of function, meaning the interpretive part slows them down but adds features not found in compiled code.
• Scripting also allows a great deal of interactivity with the content of Web pages.

Scripting
• The scripting language has become the go-between of browser and applets, taking parameters from the browser and passing them into the applet to control execution.
• The way the browser compartments this function is illustrated in Figure 3.4.

JavaScript
• JavaScript is a language designed to be placed entirely in Web pages. JavaScript is meant to be interpreted as it is read by the browser, but is executed in response to user actions.
• Figure 3.5 is a short example of JavaScript. This script causes a menu to appear when the mouse is moved over the menu title. The menu disappears when the mouse is moved off the title.

JavaScript
• The first <div> statement creates the division that the browser keeps track of so that when the mouse enters its display area, the event handlers will be activated.
• The second <div> tag is used to create an arbitrary division called “professional”.
• The result is shown in Figure 3.6.

**Components**

• Components are programs that are not standalone, but are routines that can be called upon by other programs to perform a specific task.
• Components must be written in a specified way to qualify as callable; so their methods can be called by any other program.

**Components**

• An applet is a component, but so is the Java VM. These are both programs that run inside of and which are at the service of other programs.
• The applet runs inside the Java VM, and the Java VM runs inside the browser.
• They are not just components. Because they are also objects, they are *component objects*.

**Applets**

• An applet is an executable Java program in a Web page.
• The difference between an applet and full-fledged Java program is only the security limitations placed on the applet by the Web Browser.
• The only reading and writing an applet is permitted to do is to files on its home server.

**Applets**

• This way, an applet can be used to look up database records at a central site and display the results in its applet window.
• This allows us to program database inquiries, write interactive game programs or anything else.
• There are people out there working on cryptographic solutions to provide security.
• This restricted area in which applets are allowed to play is called the *sandbox*.

**ActiveX Controls**

• ActiveX is a specification from Microsoft for components.
• The primary differences between an applet and ActiveX control are:
  – Applets are always written in Java, Visual Basic, or anything else that will run on the user’s computer.
  – Applets are downloaded and then run under the Java VM and are discarded afterwards.

**ActiveX Controls**

• The primary differences between an applet and ActiveX control are (continued):
  – ActiveX controls are downloaded and actually installed on the user’s computer and afterwards they are available to the Web browser or any other application that wants to use them.
  – Applets run in the sandbox and cannot do anything with data on the user’s computer.
  – ActiveX controls have free rein.

**ActiveX Controls**

• ActiveX controls can do this because of increased security.
• Every ActiveX control has a sophisticated class ID number that identifies where it came from.
• The drawback to ActiveX controls is that they are executables that have to be compiled for the user’s specific OS.

**Object-Oriented Programming**

• Von Neuman programming is characterized by sequential processing.
• Structured programming involves breaking code up into chunks that can be more easily managed than a program full of GOTO statements.
• A perfect structured program would contain no GOTO statement.

**Object-Oriented Programming**

• Object-oriented programming is a way of thinking of code blocks as actual physical objects.
• Object features:
  – An object is still a program but we can look at the program from the outside in.
– An object is a program that contains both data and behavior (method).

**Object-Oriented Programming**

- Object features (continued):
  – Objects are designed to be inserted in statements as if they were variables.
  – Object-oriented programming offers the opportunity to easily access huge libraries of external subroutines.

**Markup Language**

- Markup language is a static descriptive language.
- It is interpreted by the browser when it is first read in, and the results are displayed in the browser window.
- Markup is information added to text. Markup can tell what the text means and gives information on how the text is to be interpreted by the human reader or display program.

**Internet Programming**

- The Web was originally programmed with markup languages to display static text and images.
- Next came animated images. The release of the Java language allowed developers the freedom to write browser plugins that would show multimedia.
- ActiveX controls were created to increase the functionality of downloaded components.

**Internet Programming**

- Even though we can display moving images and launch applets and ActiveX controls in our page, the page content is still static.
- Dynamic HTML addresses these concerns.
- XML is a metalanguage to let Web users design their own markup language.

CS 898N – Advanced World Wide Web Technologies

**Lecture 5: HTML, XML, SGML**

Chin-Chih Chang

chang@cs.twsu.edu

Markup Language
• Markup languages evolved out of a desire to display text in something other than a single font and type size.
• Terminals advanced from one-line-at-a-time style to a text page display with the ability to place the cursor in a specific character position.
• In 1990s the Macintosh and Windows operating system bring us software to create electronic documents.

**Markup Language**

• Soon increasingly sophisticated typesetting and page layout programs became available.
• There are two kinds of markup languages:
  – the control code markup that characterize typical word processing and page layout applications in the form of embedded property symbols that are not human readable;
  – HTML-style markup using plain text characters that are both human and machine readable.

**Markup Language**

• Markup languages add processing information to text and store the combination in a file that is meant to be read by a computer.
• Markup is extra information placed with text to describe how the text is to be interpreted.

**Markup Language**

• Interpretation can be accomplished by a computer program such as a Web browser for display purposes, by an information storage and retrieval system (which includes cataloging/indexing and search programs), or by a system that does both.
• Word processing programs use binary codes that are not human readable. Hypertext markup languages use human-readable codes in plain text.

**Markup Language**

• HTML is all about looks, or *format*, which is the computer term for the way electronic information is presented.
• The most compelling reason to add markup to a document is to give it a structure so that all of its textual components can be
identified and given meaning beyond how it will appear.

**Markup Language (Example)**

```xml
<book>
  <booktitle>
    Fast Track Guide to Web Programming
  </booktitle>
  <author>
    by David Cintron
  </author>
  <image src="fast-Web-programming.jpg"
  <publish>
    ISBN 0-471-32426-4
    400 pages
    January, 1999
  </publish>
</book>
```

**Markup Language (Example)**

- This page includes four elements:
  - Book title
  - Author
  - A graphic of the textbook
  - Publishing information
- We have split each piece of information out into an element identifiable by human or machine. This format could easily be read by a search cataloging program.

**Markup Language (Example)**

- This format could easily be read by a search cataloging program, and used by another program to apply specific formats to each type of item.
- These items could be read from a database and built on-the-fly into this type of document, or this document could even serve as a database itself.
- This sample shows the idea of a markup language. The HTML file is shown in the next page.

**Markup Language (Example)**

```html
<html>
<head><title>Fast Track Guide to Web Programming</title></head>
<body>
</body>
```
Markup Language

- Documents written in languages such as HTML are becoming popular because corporate intranets are steering office communications towards paperless markup document.
- Presentations including slides, pictures, even audio and video files can be written and delivered electronically without having put materials in binders.

SGML

- SGML (Standard Generalized Markup Language) is a standard for how to specify a document markup language or tag set.
- Such a specification is itself a document type definition (DTD). SGML is not in itself a document language, but a description of how to specify one.
- SGML is based somewhat on earlier generalized markup languages developed at IBM, including General Markup Language (GML) and ISIL

SGML

- SGML is based on the idea that documents have structural and other semantic elements that can be described without reference to how such elements should be displayed. The actual display of such a document may vary, depending on the output medium and style preferences.
- Some advantages of documents based on SGML are:
  - They can be created by thinking in terms of document structure rather than appearance characteristics (which may change over time).
– They will be more portable because an SGML compiler can interpret any document by reference to its document type definition (DTD).
– Documents originally intended for the print medium can easily be re-adapted for other media, such as the computer display screen.

SGML and DTD

• SGML is extremely sophisticated.
• The language that this Web browser uses, Hypertext Markup Language (HTML), is an example of an SGML-based language.
• A document type definition (DTD) is a specific definition that follows the rules of the Standard Generalized Markup Language (SGML).

DTD

• A Document Type Definition is an exact specification for the structure of documents written in SGML.
• In order to be effectively processed, all of the elements contained in the document must be described within the DTD.
• The HTML language is described by specific SGML DTDs. But browsers do not care about HTML DTDs, and most pages don’t even have a DTD declaration.

DTD

• The browsers always process the Web pages against the latest HTML version.
• IBM and many large and small corporations are converting documents to SGML, each with its own company document type definition or set of definitions.
• For corporate intranets and extranets, the document type definition of HTML provides one new "language" that everyone can format documents in and read universally.

XML

• The XML (eXtensible Markup Language) is designed to deliver SGML information over the Web while overcoming the limitations of HTML.
• XML is a metalanguage to let Web users design their own markup language.
• XML is a simplified form of SGML which embraces the Web ethic.

XML
• XML has almost all of the capabilities of SGML but those that primarily affect document creation.
• XML, a formal recommendation from the World Wide Web Consortium (W3C).

Writing HTML Documents
• You can use a Web page editor to write HTML documents. But looking at HTML code lets you know your options and be able to debug and stretch HTML to its limits.
• Examples of Web page editors are:
  – AceHTML 4, Arachnophilia, EasyHTML, Evrsoft 1 Page
  – Netscape Composer, Microsoft FrontPage, Adobe Golive, Macromedia Dreamweaver

Writing HTML Documents
• In HTML a tag is a command to the browser to display or otherwise process the contents of the tag set in a specific way.
• An HTML element may include a name, some attributes and some text or hypertext, and will appear in an HTML document as
• A tag can also include attributes, which supply additional information about the content to be processed.

Writing HTML Documents
<tag_name attribute_name=argument> text </tag_name>
• Users should be aware that HTML is an evolving language, and different World-Wide Web browsers may recognize slightly different sets of HTML elements.
• For general information about HTML including plans for new versions, see http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html
• An HTML document is divided into two main sections: head and body.

Writing HTML Documents
• HTML begins with the tag <html>.
• A basic empty HTML document would contain these elements:
```html
<!doctype HTML public
    "DTD Specification">
<html>
<head></head>
<body></body>
</html>
```

Writing HTML Documents

• These elements are all optional. The browser will display a page just the same without any of these tags.
• Documents would be more structural with these tags. There are advantages to including these tags, such as adding more tags that go within the `head` tag.
• The `head` section contains basic information about the document, including its title and a description of its contents in the form of `meta` tags.

Writing HTML Documents (Head Element)

• The content of the `meta` tags was probably originally designed for human consumption but has ended up being used mainly as fuel for search engine indexing robots.
• `<head>` elements include:
  – Title: This tag specifies what is displayed at the top of the browser window. Search engines also use this tag as the title they show for your page.
  – Meta: This tag is for search engines and has two attributes: `name` and `content`.
    – `Meta name = "keyword" "description"`: Depending on what algorithms the search engines are using, the “keywords” and “description” attributes will play a part.
    – `Meta content = "keywords"`: The phrases in this attribute must be separated by commas.
    – `Meta content = "description"`: A good concise description of your page will go far with search engines.
Writing HTML Documents (Head Element)
• The following code from the www.prolotherapy.com homepage is an example of meta tags.
<HEAD><TITLE>Prolotherapy.com home page</TITLE>
<META NAME="keywords"
CONTENT="prolotherapy, arthritis, back pain, sports injury,
non-surgical treatment, chronic pain">
<META NAME="description"
CONTENT="a comprehensive information database on Prolotherapy, a non-surgical and permanent treatment for chronic pain">
</HEAD>

Writing HTML Documents (Body)
• The body tag is where we do all the work in HTML.
• HTML BODY attributes have:
  – background = “image”: This defines the background image for the page.
  – bgcolor = color: This gives a color to the background.
  – text = color: Specifies the body text color.

Writing HTML Documents (Body)
<meta http-equiv="refresh" content="30; url=http://www.californiado.org/aopsc.htm”>
• The original purpose of a meta tag was to give specialized information about the document to an application accessing it so the application could make an informed decision about what to do with it.

Writing HTML Documents (Body Element)
• Text Elements:
  – <p> indicates a new paragraph.
  – <pre> . . . </pre> identifies text that has already been formatted (preformatted) by some other system and must be displayed as is.
  – <blockquote> . . . </blockquote> include a section of text quoted from some other source.

Writing HTML Documents (Body Element)
• Physical Styles:
  – b: Display text in bold. <b>Buy now!</b>
  – i: Display text in italics. <i>Try again!</i>
  – u: Display text underlined. <u>Notice!</u>
  – s: display text with strikethrough. <s>Ah!</s>
• Headers:
  – <h1> . . . </h1> Most prominent header
  – <h2> . . . </h2>

**Writing HTML Documents (Body Element)**

– <h3> . . . </h3>
– <h4> . . . </h4>
– <h5> . . . </h5>
– <h6> . . . </h6> Least prominent header

• Logical Styles:
  – <em> . . . </em> Emphasis
  – <strong> . . . </strong> Stronger emphasis
  – <code> . . . </code> Display an HTML directive

**Writing HTML Documents (Body Element)**

– <samp> . . . </samp> Include sample output
– <kbd> . . . </kbd> Display a keyboard key
– <var> . . . </var> Define a variable
– <dfn> . . . </dfn> Display a definition (not widely supported)
– <cite> . . . </cite> Display a citation

• Hypertext Linking
  – <a name="anchor_name"> . . . </a> Define a target location in a document

**Writing HTML Documents (Body Element)**

– <a href="#anchor_name"> . . . </a> Link to a location in the base document, which is the document containing the anchor tag itself, unless a base tag has been specified.
– <a href="URL"> . . . </a> Link to another file or resource
– <a href="URL#anchor_name"> . . . </a> Link to a target location in another document

**Writing HTML Documents (Body Element)**

– <a href="URL?search_word+search_word"> . . . </a> Send a search string to a server. Different servers may interpret the search string differently. In the case of word-oriented search engines, multiple search words might be specified by separating individual words with a plus sign (+).
The structure of a Uniform Resource Locator (URL) may be expressed as: resource_type:additional_information

A more complete description of URLs is presented in http://www.w3.org/addressing/

Writing HTML Documents (Body Element)

- Special Characters (Entities)
  - &keyword;
    Display a particular character identified by a special keyword. For example the entity &amp; specifies the ampersand ( & ), and the entity &lt; specifies the less than ( < ) character. Note that the semicolon following the keyword is required, and the keyword must be one from the lists presented in: http://www.w3.org/MarkUp/html-spec/html-spec_9.html

Writing HTML Documents (Body Element)

- &amp;ascii_equivalent;
  Use a character literally. Again note that the semicolon following the ASCII numeric value is required.

- List in HTML
  - Ordered list: <ol>
    <ol>
    <li> First item in the list
    <li> Next item in the list
    </ol>

Writing HTML Documents (Body Element - List)

- Unordered list: <ul>
  <ul>
  <li> First item in the list
  <li> Next item in the list
  </ul>

- Menu list: <menu>
  <menu>
  <li> First item in the menu
  <li> Next item
  </menu>

Writing HTML Documents (Body Element - List)
– Definition list: <dl>
<dl>
<dt> First term to be defined
<dd> Definition of first term
<dt> Next term to be defined
<dd> Next definition
</dl>
</dl>

Writing HTML Documents (Body Element - List)

– Directory list: <dir>
<dir>
<li> First item in the list
<li> Second item in the list
<li> Next item in the list
</dir>
</dir>

Writing HTML Documents (Body Element - Table)

• To create a table, we start with the tag table.
• The table tag takes a width attribute, which can be set as a percentage of screen width (making the table size according to the user’s screen settings), or as an actual number of pixels.

Writing HTML Documents (Body Element - Table)

• Table rows and columns are constructed using the element tr at the start of each row, and within each row a series of one or more td elements for each column.
• Row and column elements can be expanded using the rowspan and colspan.
• You can set the width of each element by using the width attribute.

Writing HTML Documents (Body Element - Table)

• Table attributes:
Align= Controls alignment of content of table. “left, right, center, justify”
-Bgcolor= Sets background color for the whole table.
-Border= Sets a border for your table and its cells. # of pixels; “0” removes any border
-Bordercolor=
-Cellspacing= sets spacing between cells # of pixels

Writing HTML Documents (Body Element - Table)

• Table attributes:
  – Cellpadding= sets padding around the content of each cell   # of pixels
  – Width= sets width for the table # of pixels or percent
• Individual Cell Attributes:
  – Align= Controls alignment of contents of cell. “left, right, center, justify”
  – Bgcolor= Sets background color for the cell.

Writing HTML Documents (Body Element - Table)

– Colspan= Spreads cell over multiple columns. # of columns
– Rowspan= Spreads cell over multiple columns. # of rows
– Valign= Sets vertical alignment. “top, middle, bottom”

• The font tag in HTML has three attributes:
  – Color= sets font color
  – Face= sets font face Any available font
  – Size= sets font size +n, n, -n

Writing HTML Documents (Images)

• The img has three attributes:
  – src=“image file url” gives you the image filename and location.
  – The set of height= and width= attributes specify the exact size of the image.
– alt = specifies a string of text to display in place of the image while it is loading.

• The img attributes are listed in table 4.12.

Writing HTML Documents (Frames)
• Frames divide the screen into sections.
• Example:
  <frameset cols="22%, 78%">
  <frame src="frameleft.html" name="frameleft" scrolling=yes>
  <frame src="frameright.html" name="frameright"
             scrolling=yes>
  </frameset>

Writing HTML Documents (Forms)
• The form tag specifies a fill-out form within an HTML document. More than one fill-out form can be in a single document, but forms cannot be nested. <form action="url"> ... </form>
• The attributes are as follows:
  – action gives the name of the script the data is to be sent to for processing.

Writing HTML Documents (Forms)
– method gives you how it is to be sent. Which method you use depends on how your particular server works; we strongly recommend use of (or near-term migration to) post. The valid choices are:
  - get - this is the default method and causes the fill-out form contents to be appended to the URL as if they were a normal query.
  - post - this method causes the fill-out form contents to be sent to the server in a data body rather than as part of the URL.

Writing HTML Documents (Forms)
– enctype specifies the encoding for the fill-out form contents. This attribute only applies if method is set to post.
• Example:
  <form action="cgi-bin/fmail.pl" method="post">
    <input type="submit" name="submit1">
    <input type="reset" name="reset1">
  </form>
Writing HTML Documents (Forms)

• These two specific input type statements use the HTML keywords *submit* and *reset*.
• The *submit* button wraps up the content and sends it to a PERL script called fmail.pl.
• The *input* tag creates boxes for input.
• There are several types of input we can ask for. Type=*hidden* input is information we want sent along with the form that the user does not see or enter.

Writing HTML Documents (Forms)

• The *name* and *value* field pairs are sent to the script.
• *type = text* input creates the simple visible text box.
• *type = password* input works the same way as *type = text*, indicating only stars to the user.
• *type = radio* input creates a bullet selection.

Writing HTML Documents (Forms)

• *type = checkbox* input creates a little box to check.
• The *textarea* gives a two-dimensional area for text entry. It has the necessary name attribute and *rows=* and *cols=*, which specify the dimensions of the box in character units.

Writing HTML Documents (Forms)

• The *select* tag creates a static or pull-down list of multiple items. For each selection in the list we have the *option* tag.

Project Components

• Database connectivity
• Multimedia
• Flexibility – adapt to distributed computation
• Security
• Client-side - some client-side computation

Project Schedule

• **Sep. 5** Team composition & basic idea
• **Sep. 24** Rough plan & implementation requirements due
• **Oct. 29** Status report (<1 page, email)
Nov. 26 - Dec. 7 Oral project reports (rough draft of written due 2 days prior to talk)
Dec. 9 Final report due by noon. Electronic submission is required, in Postscript, PDF, or Word format.

Coming next

- Perl and CGI
- Project Guideline
- Program Guideline
- Working examples on Windows and UNIX
- Maybe Homework 1

CS 898N – Advanced World Wide Web Technologies

Lecture 6: PERL and CGI
Chin-Chih Chang
chang@cs.twsu.edu

PERL and CGI (PERL)

- PERL (Practical Extraction and Report Language) is a text processing language that runs in the background on servers to deliver up Web content in a fashion that is invisible to the viewer.
- PERL was invented by a man named Larry Wall as as much improved version of awk.

PERL and CGI (awk)

- awk is a standard UNIX command set and is an advanced text processing utility used for text search and replacement on a large scale.
- A, w, and k are the initials of Aho, Kernighan and Weinberger who wrote the first version of awk back in the 1970s.

PERL and CGI

- PERL was around before JavaScript, VBScript, and Active Server Pages, and was initially responsible for all programming enhancement to HTML.
- CGI (Command Gateway Interface) is the interface offered by a
Web server to pass on form data to an external application. The application processes the data and often sends back the results to the client browser.

PERL and CGI

- PERL is used to write CGI scripts. CGI scripts are the programs that process the information submitted in HTML form submissions.
- Forms can be used for anything from spawning an automatic e-mail to database searches and electronic storefronts. The power of PERL can support any and all of these activities.

Download PERL

- Freeware versions of PERL are available at [www.perl.com](http://www.perl.com).
- There are two ways to install PERL:
  - Download the source and compile it.
  - Download the binaries (executables) and run the install batch.
- To download the binaries, go to [reference.perl.com](http://reference.perl.com).

Installing PERL for Windows

- To download PERL for Windows:
  - Select the Win32 freeware version. The ActiveState version is also available and has a load of extra features but it is not free.
  - Download `perl5.00402-dist04-bc.zip` and `libwin32-0.16.zip`.
  - Unzip the perl5 file, then the install.bat under DOS. Add the PERL/bin directory to your path as instructed in the readme.win32 file. Then unzip the libwin file and run its install.bat.

CGI Works with PERL

- The problem that CGI solves is what to do with user input from an HTML form. The solution has been programmed into HTML by means of the `form` tag combined with the `submit` button. The `submit` button causes two specific actions to occur:
  - The data from the form is placed into `environment variables`.
  - The CGI script named in the form tag `action = "scriptname"` parameter is executed. CGI scripts are written in PERL.

CGI Works with PERL

- The contents of any and all environment variables are available
within PERL and used as textual data in PERL statement.

- The way to access these variables is to write a reference as below:
  \$ENV(variables)
- The form tag method parameter specifies either get or post as the mechanism through which the user input is transferred to the environment variables.

**GET and POST**

- Both methods format the user input date in the same way. The difference between them is in how that data is retrieved by PERL.
- HTML form input consists of several options for user input: the text box, radio button, checkbox, and select group.
- User input from these options is paired up in the format name=value.

**GET and POST**

- Spaces in the value side are placed with + symbols and the pairs.
- Here is an example:
  name=Bill+Gates&company=Microsoft
- This string would be generated by a form with two text input boxes.
- The get and post methods need a CGI script to send the data to.

**GET and POST**

- The form tag has its method and action parameters as follows:
  \(<\text{form method="get" action="cgi-bin/signup.pl"}>)\n- The .pl extension means this is a PERL script and \text{cgi-bin} is the directory where these are usually located.
- In UNIX, bin is short for binary. Here we have a form that will format the user input, load it into environment variables, and transfer control to \text{cgi-bin/signup.pl}.

**GET and POST (The Get Method)**

- Get is the default CGI method so if no method is specified, get will be used.
• When the get method is executed, the CGI data is formatted in two pieces, separated by a “?”.
  – The contents of the action=parameter
  – The formatted user input
• This is then used to set the next URL for the Web browser, causing control to be transferred to the script.

GET and POST (The Get Method)
• The formatted user input is also placed in the environment variable named QUERY_STRING from where the PERL script can read it. This is limited to 1024 characters. This would look like:
cgi-bin/signup.pl? name=Bill+Gates&company=Microsoft
• The user input shown in the preceding code would give this result.

GET and POST (The Post Method)
• The form tag in the post method would look like this:
  <form method="post"
    action="cgi-bin/signup.pl">
• Post directs the formatted user input into standard input buffer, which is commonly called stdin.

GET and POST (The Post Method)
• For PERL to access this, it opens stdin and reads the line.
• This data get transferred from the browser form elements to the server through HTTP.
• In the post method, the content-length and other environment variables are included in the header.

GET and POST (The Post Method)
• There is no limit to the size of the post string.
• In the CGI string of the post method special characters that have specific meaning in PERL are encoded as a hexadecimal value preceded by a % sign.

CGI Input Controls
• There are various types of input controls in the form method.
• Given an example of buying a car, we might have the following
input types.

• The text box:
  
  `<input type=text name=model>
  – If the user enters porsche boxster, it will be returned as model=porsche+boxster.

  **CGI Input Controls**

• The radio button:
  
  `<input type=radio name=color value=silver checked>
  `<input type=radio name=color value=gold>
  `<input type=radio name=color value=red>
  – The radio box uses the value given in the input tag, returning color=silver.

  **CGI Input Controls**

• The checkbox:
  
  `<input type=checkbox name=tires value=goodyear>
  `<input type=checkbox name=tires value=bose>
  – If nothing is checked, nothing is returned. If no value is specified, true will be used. This set could return tires=goodyear.

  **CGI Input Controls**

• The select box:
  
  `<select multiple name = extras>
  `<option>air conditioning
  `<option>leather seats
  `<option>leather seats
  `<option value=gps>global positioning system
  </select>

  **CGI Input Controls**
  – The select box will return 0 or more pairs for options selected. If no value parameter, the value returned will be the text of the option.
  – The select box multiple attribute permits multiple selections. If more than one option is selected there will be only be one pair returned, but
the value side will have all the selections separated by “\0”. The selections in the preceding example would return extras=air+conditioning\0leather+seats\0gps.

**CGI Input Controls**

- The submit button:
  
  ```
  <input type=submit name=“Buy the car!”
    value=buyit>
  ```

- The submit button will be included as well and will give you a Buy+the+car%21=buyit.

**Writing Perl Scripts**

- Perl is an interpreted language.
- Perl scripts, like scripts in other interpretive languages, are compiled on the fly when they are run.
- Though there are tons of Perl resources online, we have to modify them to fit our particular domain name, e-mail addresses, our host’s directory structure, and whatever files we are accessing on the server.

**An Example CGI/Perl Translation**

- We use the HTML in Figure 9.4 as an example.
- This form contains three tables:
  - one for the user identification
  - one for the product selection
  - one for the credit card input
- The CGI action is given in this form tag, which starts near the beginning and ends at the bottom of the page.

**An Example (HTML Form)**

```html
<form action=“cgi-bin/fmail.pl” name=“orders” method=“post”>
  There are three hidden input fields declared: the e-mail recipient, the subject, and the URL for the CGI script to link to after it has sent the e-mail
```
Perl - Elements

- The language elements of Perl fall into four categories:
  - Data types: scalars, arrays, and hash arrays
  - Statements
  - Regular expressions
  - File operations

Perl – Data Types

- The basic Perl data types are:
  - Scalar - $data
  - Array - @data
  - Hash array - %data
  - File handle – data

- Perl variable names are case and type sensitive.

Perl – Data Types (Scalar)

- The variable name must start with a letter and may contain any combination of letters, numbers, and the underscore _ character.

```perl
$scalar - $string, $number, $array[$n], $hash_array{$n}
```

- Numeric types are internally represented as double-precision floating-point.
- Numeric can be assigned as integers, fixed-point, or in floating-point notation.
- For example, 186282 can be written as 1.86282e5.
- String types are enclosed in either ‘single-quote pairs’ or “double-quote pairs”.

Perl – Data Types (Scalar)

- Strings in single quotes are true literals and use a backslash for
an escape key, \ as a single quote, double backslashes \ as a backslash.
• Double-quoted strings support a Perl string replacement function called variable interpolation.

Perl – Data Types (Scalar)
• Any variable name found inside a double-quoted string will be replaced by the value of that variable.
• For example, the first two lines would result in the third line being printed:
  $name = “Larry Wall”;
  Print “The value of name is $name.”;
  The value of name is Larry Wall.

Perl – Data Types (Scalar)
• Variable interpolation can be avoided by breaking up the string into parts using the “.” string concatenation operator.
• For example,
  ‘The value of $name is ’ . $name; # or
  ‘The value of $’ . ”name is $name”;
• If a variable is used but not defined, a blank or 0 value is assigned.

Perl – Data Types (Scalar)
• If $name was not defined, the preceding statement would not result in an error but in an empty string as follows:
  The value of $name is
• Double-quoted strings support a full range of escaped characters including \ (backslash), ‘” (double quote), \cX (any control character; e.g., control-x), \e (escape), \n (newline), \t (tab), and \xFF (any hexadecimal character).

Perl – Data Types (Array)
• The most common use of the escaped character is the newline in the print function.
  print “Hello World\n”;
  @array - @array
• A list (array) is a collection of scalar data in a specific order, and an array is a variable that holds a list.
• A list can contain both string and numeric values.

Perl – Data Types (Array)
• () represents an empty list with no elements.
• A list literal may be composed of a mixture of numeric and string values, scalar variables, ranges (..), and even other lists.
• In the following code, the quote word (qw) function is used to create arrays without typing quote-marks.

```
@rgb = qw(red green blue)
@cmy = qw(cyan magenta yellow)
```

Perl – Data Types (Array)
• The following array is formed from a list of two scalar literals and two arrays.
```
@colors = ("black", "white", @rgb, @cmy);
```
• The list constructor operator ".." can be used to create an integral range of numeric values.
```
@floors = (1..12,14..22);
$x = .5; $y = 9.5;
```

Perl – Data Types (Array)
```
@halfsizes = (.5..9.5);
@halfsizes = ($x..$y);
```
• Perl is very flexible in how lists can be used on both sides of the equal sign. These statements are all valid ways to assign values:
```
($red, $green, $blue) = qw(red green blue);
($top, $bottom) = ($bottom, $top);
```

Perl – Data Types (Array)
• Array elements are referenced using the [] bracket pair, which may contain a literal, scalar variable, or expression.
• The following statement switches two array values by using two lists referencing the same array.
```
($name[$first],$name[$last])= ($name[$last],$name[$first]);
```
Perl – Data Types (Array)

• The following one accomplishes the switching by using a technique called slicing. A slice of the array is represented by using the format $array[element list].
  @name[$first, $last] = @name[$last, $first];
  @name[0, 2] = qw(Larry, Wall);
• The highest element of an array is represented by the $#array variable.

Perl – Data Types (Array)

• In Perl, the last element can be indexed by -1.
  $ants[$#ants] = “little one”;
  $ants[-1] = “little one”;
• Arrays can also be assigned to each other directly and by combing array slicing with array assignment.

Perl – Data Types (Hash Array)

@array2 = @array1;
@slicearray = (0, 2, 4, 6, 8);
@array3 = @array2[@slicearray];
%hash_array - %hash_array

• A hash array is a list of string keys and values that are paired together.
  $partridge = “a partridge in a pear tree”
  %christmas = ($patridge, 1, “turtule doves”, 2, “french hens”, 3);

Perl – Data Types (Hash Array)

• Individual hash array elements are assigned by using the string key in place of the element number.
  $christmas{“calling birds”} = 4;
• The following two lines should read and then restore the hash array with no changes.
  @twelve_days = %christmas;
  %christmas = @twelve_days;

Perl – Data Types (Hash Array)

• The following line creates a new hash from three pairs converted
from the @presents array.
%
favorites=@presents[6,7,15,16,17,18];

• The following two lines copy hash arrays.
%
next_year = %christmas;
%
allpresents = (%mypresents, %yourpresents);

Perl – Data Types (Hash Array)
• Hash keys and values can be extracted separately using the keys and values functions.
%
@presents = keys(%christmas);
@
days = values(%christmas);

• The defined() function returns a Boolean value based on whether the scalar or array element has had value assigned to it.
%
if (defined(%presents("new car"));

Perl – Statements (Operator)
• Perl supports an extended set of mathematical and string handling operators, and a standard set of comparison operators.

– +-* / : add, subtract, multiply, divide
– ** : exponentiation, $e = m*c**2.
– % : modulus, 10%3 = 1.
– ++ -- : auto- increment and decrement.
– $x : string replication, $test = test x 3.
– .. : string concatenation,

Perl – Statements (Operator)
– .. : range, 1..5 = 1, 2, 3, 4, 5.
– =~ : match. Used with regular expressions for string search and replacement, $val =~ tr/+/ /;
– !~ : no match.
– & & and || : logical AND and logical OR.

• Perl has separate comparison operators for numeric (==, !=, <, <=, >, >=) and string (eq, ne, lt, le, gt, ge) variables:

– Equals: ==, eq
– Not equal: !=, ne

Perl – Statements (Operator)
– Less than: <, lt
– Less than or equal: <=, le
Greater than: \( > \), \texttt{gt}  
Greater than or equal: \( \geq \), \texttt{ge}

- **String manipulation functions:**
  - \texttt{split()}: Split returns a list of strings by split from a single string at the operator expression.
    
    
    \$cgiinput = 'http://sun.com';  
    @pairs = split (/:/, $cgiinput);  
    $pairs[0] = "http"; $pairs[1] = "/sun.com"

  - **Perl – Statements (String Manipulation)**
    - \texttt{join()}: Join returns a single string from a list of strings joined together by the glue string.
    
    
    @statements = qw (Hello World Welcome);  
    $string = join (", ", @statements);  
    $string = "Hello, World, Welcome";
  - \texttt{sort()}: It does an alphanumeric sort.
  - \texttt{reverse()}: It reverses the elements.

- **Perl – Statements (Array Manipulation)**

  - \texttt{push()}: It adds an element into the end of the array.
  - \texttt{pop()}: It removes an element from the end of the array.

- **Perl – Statements (Hash Array Manipulation)**

  - \texttt{delete()}: It is used to remove hash array elements.
    
    
    delete @nominees ("the avengers");
  - \texttt{each()}: It provides easy looping through hash array pairs.
while (($title, $star) = each (%movie)) {
    print "The star of $title is $star\n";
}

Perl – Statements (if)

• There are two types of if statements
  – if..else and elseif
    if (expression) {statements};
    if (expression) {statements}
    else {statements};
    if (expression) {statements}
    elseif {statements} else {statements};
  – unless..else
    unless (expression) {statements};

Perl – Statements (if)

unless (expression) {statements}
  – Perl supports the following extension:
    {statements} if (expression);
    {statements} unless (expression);

Perl – Statements (Loop)

• Loop statements:
  – while (expression) {statements}
  – until (expression) {statements}
  – do {statements} while (expression)
  – while (expression) {statements}
  – until (expression) {statements}
  – do {statements} while (expression)

Perl – Statements (Loop)

  – do {statements} until (expression)
  – for (initial value; test condition; increment) {statements};
  – foreach $scalar (@array) {statements};

• There are several statements that modify the sequence of loop execution when placed inside the loop. These include last,
next, and redo.

Perl – Statements (Loop)

- There are several statements that modify the sequence of loop execution when placed inside the loop. These include:
  - `last` breaks out of the loop that contains it.
  - `next` skips the remainder of the loop and start at the next increment.
  - `redo` goes back to the start of the loop without incrementing.
- These statements are usually part of an `if` or `else` clause.

Perl – Statements (Loop)

- When you have a loop within another loop the last, next, and redo statements by default apply to the innermost loop.
- They can apply their action to an outer loop if that loop has a label.
- The following code shows the power of these statements.

```perl
LOOP:
while (expression) do {
    statements
    if (condition) { last }
}
POOL:
until (expression) do {
    if (condition) { next LOOP }
    statements
}
Perl – Statements (Loop)
for ($I = 0; $I < $j; $I++) {
    if (condition) { redo POOL }
    if (condition) { next }
    statements;
}
```

Perl – Statements (Subroutines)

- Subroutines are defined with the `sub` `subname` statement.
• Subroutines can be called in either of two ways:
  – prefixing the subroutine name with an & sign as in &subname.
  – subname() call format.
• Subroutines can return values so that subroutine calls can be
  used in assignment statements, as in $string = subname ()

**Perl – Statements (Subroutines)**

• Take the following Warp Drive Calculator as an example:
  
  ```perl
  $traveltime = &warpdrive;
  $triptime = warpdrive();
  sub warpdrive {
      $warpfactor = 2 ** ($warp – 1);
      $traveltime = ($distance / $warpfactor) * 365.25;
      return $traveltime
  }
  ```

**Perl – Regular Expressions**

• An expression is a group of symbols that gives a result.
• Regular expressions in Perl are used for pattern matching.
• Regular expressions can be used in two types of statements:
  assignment and comparison.
• There are three components to a statement with a regular
  expression:

  **Perl – Regular Expressions**

  – **The regular expression** is usually contained in a forward slash //
    symbol pairs or m followed by the preferred symbol, for example,
    /target/ and m!target!.
  – **The source string** is the string that will be searched for the
    expression. regular expression.
    + If the statement is a comparison, pattern matching will return a true
      or false result.
    + If the statement is an assignment, the source string will have pattern
      matched substrings found in the string.

  **Perl – Regular Expressions**

  – **The operator** binds the source string to the regular expression.
    + Only the =~ match operator can be used in an assignment
      statement.
Both the =~ match operator and the !~= not-match operator can be used in comparison statements.

The following match operator is a string replacement.

```perl
$value =~ tr/+//;
```

Perl – Regular Expressions

The following line is using not-match operator to skip lines that begin with a comment mark.

```perl
if ($string !~ /^#/) {.. Statements ..}
```

- These are Perl pattern matching options:
  - abc : match the character string “abc”.
  - . : match any character except newline.
  - a? : match zero or one instance “a”.
  - a+ : match one or more repetitions of “a”.

Perl – Regular Expressions

- a* : match zero or more repetitions of “a”.
- a{2} : match exactly two repetitions of “a”.
- a{2, 4} : match between two and four repetitions of “a”.
- a{4,} : match four or more repetitions of “a”.
- ^ : match beginning of line, e.g., /^#/.
- $ : match end of line, e.g., /money.$/.
- [a-b] : match any character within the range a to b.

Perl – Regular Expressions

- [abcde] : match any character within the brackets.
- [^a-b] : match any character except those in the range a to b.
- [^abcde] : match any character except those within the brackets.
- a|b : match a or b.
- \ : match the next character literally.
- \d : match any digit. Same as [0-9].
- \D : match any nondigit. Same as [^0-9].

Perl – Regular Expressions

- \s : match any space character including newline, tab, return, and form feed. Same as [\n\t\r\f].
- \S : match any non-space character.
- \w : match any word character including letters, numbers, and underscore. Same as [0-9a-zA-Z].
- \W : match any non-word character.
• The amount of complexity that can result leaves many programmers scratching their heads to try and figure out what kind of string the writer was trying to match.
• The principle is to keep it simple and write comments.
• Perl statements that use pattern matching include substitution and transliteration.
  
  Perl – Regular Expressions
  • The substitution statement directly replaces instances of pattern matching with the substitution string and is formatted as s/regular expression/substitution string/options.
  • This substitution statement globally replaces all + signs with a space.
    $$\textit{value} =~ \texttt{s/+/  /g;}$$
  
  Perl – Regular Expressions
  • The transliteration statement is a simpler version that searches for expressions on a character-by-character basis and replaces them with the respective characters in the substitution string.
  • Each character can be represented by a regular expression. This statement is formatted as tr/characterlist/characterlist/options.
    $$\textit{value} =~ \texttt{tr/+ /g;}$$
  • The substitution and transliteration statements take following pattern matching options:
    – /e : evaluate right side as an expression
    – /i : ignore case in the search string.
    – /g: replace globally, find all matches.

  Perl – File Operations
  • The basic Perl file function is contained in the diamond operator <>. This acts on a file to read in the next line.
  • The default file is STDIN. So the operator alone, as in $line = <>, will return a line from STDIN.
  • The diamond operator returns a string up to and including the
line terminator which is different among systems.

Perl – File Operations

• For example, “test” is not equal to “test\n”. chop() is used to remove the last character of a string and chomp() is used to remove the last character only if it’s a newline.
• The basic Perl file function is contained in the diamond operator <>
• Basic Perl file operations are open, <>, print, and close.
• When opening a file in Perl, a filename is associated with a file handle.

Perl – File Operations

• The file handle is the only data type in Perl that has no preceding symbol.
• It is suggested that such data types, which include labels and file handles, are given uppercase names to avoid conflicts with future modifications to the Perl language. For example, if you have a file handle “data” and Perl comes out with a function “data”, your script will become defunct.

Perl – File Operations

• There are three ways to open files: for input, output, and to append data to an existing file:
  – Input: open (FILE, “<inputfile”);
  – Output: open (FILE, “>outputfile”);
  – Append: open (FILE, “>>biggerfile”);
• The lines in the following code open a file for input, read the lines into an array, close the file, and trim the newlines.

Perl – File Operations

• The default open method is input.
• The array combined with the diamond operator reads in the entire file automatically.

```perl
$filename = “data.txt”;
open(FILE, “$filename”);
@lines = <FILE>;
close(FILE);
```
chomp(@lines);

Perl – File Operations

• In the situation that the script cannot open a file, the die () function can be used.
• The die () function operates off the fact that Perl evaluates the results of an open statement to a Boolean value. We can use logical statement structure to create a error handling.
• The following statements are all equivalent.

Perl – File Operations

open (FILE, "$filename") ||
die "Unable to open $filename"
unless open (FILE, "$filename")
{ die "Unable to open $filename" }
die "Unable to open $filename"
unless open (FILE, "$filename")

• Die will print whatever is passed to it plus a system error message and then exit the script.

Perl – File Operations

• To exit with a message you can either use “exit 0” or die “\n” where the newline prevents the system error message from being printed.
• Another way to control file input in a loop is to use the while ($line = <FILE>) control loop.
• There are functions for directory:
  – opendir() – open a directory.
  – closedir() – close a directory.
  – readdir() – read a directory.
  – chdir() – change the directory.
• The following statements would open a directory, read the first file, change to the directory, and then open that file.

opendir (DATADIR, "/www/home/database");

ungefixfile = readdir (DATADIR);
closedir (DATADIR)
Perl – File Operations

```perl
chdir ("/www/home/database");
open (FILE, $nextfile);
```

- Before opening a file, you might want to make sure if it’s really a file and not a directory or something else.
- For this purpose, Perl supports a number of testing functions, all in the format:
  ```perl
  If (-flag filename)
  ```

Perl – File Operations

- These are some file test flags:
  - -d : if file is a directory.
  - -e : if file exists.
  - -s : if file exists, returns file size.
  - -z : if file exists but is zero size.
  - -T: if file is a text format file.
  - -B: if file is a binary format file.

CS 898N – Advanced World Wide Web Technologies

Lecture 10: Examples of PERL and CGI
Chin-Chih Chang
chang@cs.twsu.edu

Perl Scripts - An Example

- The Perl script of our example is `fmail.pl`.
  ```perl
  #!/usr/bin/perl
  ```
- All Perl scripts start with a line give the location of the server’s Perl interpreter.
- You can run a Perl script in this way:
  ```perl
  c:\perl\bin\perl signup.pl
  ```
- The first thing we do is to declare variables.

Perl Scripts – Declaring Variables

- There are no data types in Perl. Internally data is represented as numbers or strings.
- All numbers are double-precision floating-point. It means it is a 32-bit number.
- An array or string can be any size, up to the entire available
Perl Scripts – Declaring Variables

- Perl uses the first character to distinguish between types of variables: $ means a single number or string. @ means an array. # is the comment sign and is not a variable.
- The first line sets the value of $mailprog to “/bin/sendmail”.
  \>$mailprog = '/bin/sendmail';
- The second line fills the array @months with 12 string values.
  \>@months = (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec);
- The third line calls a Perl function localtime and fills the array @tstamp with its values.
  \>@tstamp = localtime(time);
- They are shown in the comment line following as (second, minute, hour, day of month, year, day of week, day of year, and daylight savings time flag.
  \>#tstamp = ($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst)
- The script then formats a string called $date as “dd-mmm-yy hh:mm” with the values from @tstamp.
  \>$date = $tstamp[3] . "-" . $months[$tstamp[4]] . "-" . $tstamp[5];
  \>$date .= " " . $tstamp[2] . ":" . $tstamp[1];
- When referencing single array members the non-array $ is used, so instead of @tstamp[3], it is $tstamp[3].
- Array indexes start at 0.
- The “.” operator is the concatenation sign in Perl.
- The next statement reads the contents of the message sent by the post method, which is waiting in the stdin input buffer.

Perl Scripts – Declaring Variables
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

• The read statement reads one line from the file named in the first parameter (STDIN) into the variable named in the second parameter ($buffer) up to the size named in the third parameter ($ENV{'CONTENT_LENGTH'}).

Perl Scripts – Declaring Variables
• CONTENT_LENGTH is an environment variable sent by the browser to the server through the HTTP header that delivers the CGI input.

The next section is the Perl code that breaks down the CGI name/value pairs.

Perl Scripts – Regular Expressions
• The first action is to use Perl’s split function to blow apart the string at the “&” boundaries, storing the results in the array @pairs.

@pairs = split(/&/, $buffer);
• “/&/” is a regular expression in Perl.
• A regular expression is a group of symbols that follows the rules of Perl.

Perl Scripts – Regular Expressions
• We have the expression between two slashes, and &.
• Regular expressions in Perl are used for pattern matching, where we create an expression that will match a set of characters in a character string.
• There is another regular expression below, as “%(a-fA-F0-9)a-fA-F0-9%”.

Perl Scripts – Regular Expressions
• This means the % sign followed by two elements: one instance of a character in any of the ranges a-f, A-F, or 0-9 followed by another instance of a character in the same ranges.
• This matches any instance of an encoded hexadecimal character in the format %nn.

Perl Scripts – Regular Expressions
• The statement with the `split` function takes the CGI string in `$buffer` and splits it up into several array elements using the `&` sign as the boundary between elements.
• The next block is a type of loop specific to Perl and is enclosed by a `{}` pair.
• The `foreach` statement steps through the array `@pairs`, assigning each value in turn to `$pair`.

```
foreach $pair (@pairs)
```

Perl Scripts – Loops

• Notice that each statement must end with a semicolon.
• The beginning of the block does another split, this time splitting each array element, which represents a name/value pair, on the `=` sign.
• The list (`$name, $value`) is assigned the results of the `split`.

```
($name, $value) = split (/=/, $pair);
```

Perl Scripts – Lists

• There are two types of variables: `scalar` and `list`.
• Scalar means a single value. `$variable` is called a scalar variable and hold a single value.
• A list is ordered scalar data. `@variables` are called array variables and hold lists.

Perl Scripts – Lists

• The (`$name, $value`) is a list. The expression `(1, 2, 3)` is a list literal. `()` is an empty list.
• After separating the name from the value and placing these in `$name` and `$value`, these pairs are still CGI encoded, so the next thing to do is to remove the special characters.

Perl Scripts – Removing Special Characters

• First we use the `tr`, or transliteration, function to replace each `+` sign with a space.

```
$value =~ tr/+/ /;
```

• The `=~` operator is called the substitution operator that replaces the source variable (+) with its modified self ( ).
• Next, we use the search function (s) to take the encoded hexadecimal characters and replace them with their ASCII character values.

Perl Scripts – Removing Special Characters
• For instance translating “http%3A%2F%2F” back to “http://”.
• The replacement value uses an Perl function to `pack`, into a single character represented by “c”, the hex value of $1.
• $1 represents the match found by the first regular expression in the current statement.
• The g parameter at the end means search globally, replacing all values found in the entire string.

Perl Scripts – Removing Special Characters
• The e parameter means replace using the evaluated result of the replacement string (the character), not the literal value of what is in the replacement string (“pack(“c”, hex($1))”).

```perl
$value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("c", hex($1))/ge;
```

Perl Scripts – Hash Arrays
• Next, we use a hash array to store the $name/$value pairs.
• Arrays are accessed by their element number starting with 0 (e.g., $names[0], $names[1], etc.).
• Hash arrays are given string key values and can be stored and retrieved using string keys (e.g. $surnames{“george”}).
• The hash array as a whole is referenced using the % sign.

Perl Scripts – Hash Arrays
• You would initialize a hash array using a list of the name/value pairs of the CGI scheme like this:
  ```perl
  %cgiarray=(key1, value1, key2, value2, key3, value3)
  ```
• Like regular arrays, individual hash array elements are referenced using the $ sign, but unlike regular arrays, use the {} set for index reference.

Perl Scripts – Hash Arrays
```perl
$cgiarray{$name} = $value;
```
• The next few statements in the script store the name/value pairs
in a hash array, and then use the `push` function to store them separately in two arrays: `@names` for the names and `@values` for values.

- **push** adds a value to the end of an array. **pop** takes the last array element and assign it to `$last` as in `$last = pop(@array)`.

### Perl Scripts – Hash Arrays

- There is a good reason for using the separate arrays to store the name/value pairs. We can randomly retrieve the `%cigarray` hash array values using string keys based on our CGI name parameters, but Perl decides how these are stored, which we have no control over. When we want to send the values back in the same order in which they come in, that won’t work.

```
push (@names, $name);
push (@values, $value);
```

### Perl Scripts – Errors Handling

- There is a test to see that there is somewhere to send the e-mail. This prevents us from crashing the mail program with no recipient.
- If there is none, a subroutine call to `&safe_die` is issued to display an error message.
- In Perl, the `&` symbol prefixes the name of a subroutine.

```
$target = $cgiarray{'recipient'};
if ($target eq "") {
    &safe_die("No Recipient Given\n");
}
```

### Perl Scripts – Errors Handling

- After this recipient handling, we check to see if the sender’s e-mail address was entered, and if not, replace that with a dummy
address to satisfy the requirements of the sendmail program.
if ($cgiarray{'username'} eq "") {
    $cgiarray{'username'} = "No-Email-Given\@nowhere.none";
}

Perl Scripts – Errors Handling
• The word die in Perl means to end the program.
• The primary type of error is a syntax error that will be found when you first run the script. There’s very little that can crash a Perl Program.
sub safe_die {
    print "Content-type: text/plain\n\n";
    print @_,"\n";
    exit(0);
}

Perl Scripts – Sending Mail with Perl
• First we open a file. STDIN and STDOUT are the defaults.
• STDIN comes from either the keyboard for Perl to run at the command prompt, or the CGI string for server-side Perl.
• STDOUT goes to the display for command prompt Perl or the user agent for server-side Perl.

Perl Scripts – Sending Mail with Perl
• In this example, we are opening a file and calling it MAIL. This is called a file handle.
• We are opening a pipe to $mailprog, declared at the start of the script as “/bin/sendmail”.
• The –t flag tells sendmail to read the recipient addresses from the lines labeled “To:” and “Cc:”.

Perl Scripts – Sending Mail with Perl
• The | pipe symbol means we want the output directed to the MAIL file handle to go to the input of the sendmail program, we don’t want to create a file called sendmail.
• To open a file for input, use the < symbol, and for output, use the > symbol.
• There is an or (||) symbol between the open statement and die
Perl Scripts – Sending Mail with Perl

• The result of the `open` statement is logically evaluated: If true, the second half is not executed; otherwise, it is.

```perl
open (MAIL, "|$mailprog -t") || die("Can't open $mailprog!
");
```

• Notice we that we are using variable names in the middle of strings. Perl calls this **variable interpolation**.

Perl Scripts – Sending Mail with Perl

• We can use a ‘single-quoted string’, which will not be interpolated.

• The **print** statements writes the string to `MAIL`.

```perl
print MAIL "From: $cgiarray{'username'}\n";
prompt MAIL "Reply-To: $cgiarray('username')\n";
prompt MAIL "To: $cgiarray('recipient')\n";
prompt MAIL "Subject: $cgiarray{'subject'}\n\n";
prompt MAIL "Information submitted on $date\n";
```

Perl Scripts – Sending Mail with Perl

• The name/value printing loop run through the values stored in the matched `$name` and `$value` arrays.

```perl
for ($i=0; $i<=$#array; $i++)
{
    print MAIL "$names[$i]:  $values[$i]\n";
}
```

Perl Scripts – Sending Mail with Perl

• Once this is done, the `MAIL` file is closed.

```perl
close (MAIL);
```

• The last thing is to transfer the browser location to a page that says, “Thanks for your order.”

• There is no file named in the print statement, so the default goes to STDOUT, which ends up back at the user agent.

```perl
print "Location: $cgiarray{'thankurl'}\n\n";
```
Internet Database

- The proliferation of the Internet provides an easy access to the enormous data around the world.
- To have an efficient access, an efficient information storage and retrieval techniques are required.
- The Internet database makes the proficient access available.

Internet Database Access

- Based on the concept of the client and server, Internet database is stored on a network server and the user access it through a client program.
- The current solution is to provide remote access using a client/server connection through a TCP/IP network connection using HTTP.

Internet Database Access

- The Internet or intranet provides a basis for communications. The Web browser is the client part of client/server.
- The starting point for the database service is the Web page. Through the Web page we invoke the server program, which gathers information and returns it to the client in the form of another HTTP delivered Web page.

Creating a Database

- There are two basic types of data file: **sequential** and **indexed**.
- Sequential data files are easy to maintain with the use of a plain text editor and for Perl to read through and search.
- The downside to this is that if the file is large it will slow down processing.
- The simplest and most usual form of a sequential file is a text file.

Creating a Database

- These files can also be called **flat files**, which simply means there is no index structure.
Any indexed data file is kept in a specific order based on one or more fields, and these fields combined are called the key. The key is used to access a specific record, or set the position in the file to a specific location in the file order.

Creating a Database

- Internally, indexed files use a multilevel tree structure to find data as quickly as possible.
- Each tree branch contains a list of keys and locations in the next lower level of the index where the first key in the range can be found.

Creating a Database

- Large indexed files can contain several levels of indices and several hundred thousand or even millions of records.
- The point is that an efficient index may allow direct access to any record in a database in as few as reads as possible.

Creating a Database

- If access to the data based on different fields is needed, a database can be created using more than one key, but this makes the database larger and more complex because each key needs a completely separate index.
- When you access the database, you will have to specify which key to search by. The first key used to order the file is called the primary key and all other keys are called alternate keys.

Creating an Internet Database

- To show the techniques of implementing an Internet Database, we follow the example in the textbook.
- The example builds a database containing a list of osteopathic physicians in the state of California.
- There are less than a thousand of these, and the database can be searched on any five different fields, the database is kept in a sequential file.
Creating an Internet Database

The CGI form for this example will accept up to five different fields of data to search for, including last name, specialty, city, zip code, and languages spoken.

It will then send the query string to the Perl script, which will search the respective fields for the data and return a Web page listing all doctors who match the search results.

Creating an Internet Database

The following core HTML is used to produce the form.

```html
<form name="findadoc" method="post" action="cgi-bin/findadoc.pl">
  Dr. last name: <input type="text" name="lastname">
  Specialty: <input type="text" name="specialty">
  City: <input type="text" name="city">
  Zip code: <input type="text" name="zipcode">
  Language: <input type="text" name="language">
  <input type="reset" value="start over" name="reset1">
</form>
```

Creating an Internet Database

The form uses the post method to send the query data to a script called `cgi-bin/finddoc.pl`.

Let’s say someone accesses this page and enters “prolotherapy” under specialties. The CGI engine will send an HTTP header with the following information to the Perl script:

```
lastname=&specialty=prolotherapy&city=&timezone=&language=&Submit1=Search%23
```

Creating an Internet Database (The Query)

Figure 10.5 contains the Perl script that powers this search.
• The script starts out with the standard handling for breaking down the post method CGI query string.
• The contents of the query string are read using the STDIN file handle for a length given by the environment variable CONTENT_LENGTH into variable declared on the fly called $buffer.

Creating an Internet Database (The Query)
• We then remove any newline characters of the end of $buffer by the chomp function and proceed to process the name/value pairs.
• The pairs are loaded into the array @pairs, using the split function to chop up the string at each instance of the & sign.
• Then the foreach loop iterates through the @pairs array, loading each element into the $pair variable.

Creating an Internet Database (The Query)
• Each instance of $pair is split into $name/$value variables at the = sign.
• The $name and $value variables have their + signs transliterated into a space and any instances to hexadecimal characters substituted with the actual ASCII character.
• A hash array is created using $name for the key and $value for the contents.

Creating an Internet Database (The Query)
• First the chdir statement sets our default directory to the location
of the database file.

- The the &genheader subroutine call uses the print statement to write to STDOUT.
- The genheader subroutine first writes a standard simple HTML header as shown next with two newlines following.

```
Content-type: text/html

Creating an Internet Database (The Query)
- Then the print << ‘ENDPRINT’ version of the print statement is used to output a long stream of HTML that will ended by the string ENDPRINT.
- The number of doctors found is set to 0 as $doccount, and the doctor data file is opened.
- The file handle DRS is assigned to the file californiado.dat.

Creating an Internet Database (The Query)
- The safedie subroutine prints an explicit error message, followed by whatever error message the server reports, which is represented by the Perl string “$!” in parentheses.
- The search routine is enclosed in a while control loop that repeats as long as the diamond operator successfully retrieves the next line from DRS.

Creating an Internet Database (The Query)
- Perl has two default variables, $ for scalar values and @ for array values. If no variable is specified, these are assumed.
- The split function loads the array @doc with the contents of the doctor database record retrieved by the most recent read.
- The array elements $doc[0] through $doc[11] are loaded with the values of their corresponding columns in the database.

Creating an Internet Database (The Lookup)
- The variable $found is set to 0 before we check each of five search elements entered.
- Each search criteria for that element is checked only if:
  - A search criteria for that element was entered, if (defined($form["lastname"]))
  - That search criteria is not empty, if ($form["lastname"])

Creating an Internet Database (The Lookup)

• Each search element is checked by setting the value of $found to the result of using the =~ match operator to associate the array element, $doc[n], with the contents of the search element with the case insensitive flag set.

• Finally, if $found is set, the $doccount is incremented to prevent the “No doctors …” message from being displayed, and the &genhtml subroutine is called.

Creating an Internet Database (The Result)

• The last line of the &genheader subroutine is the unordered list opening tag <ul>.

• The &genhtml subroutine writes the contents of the ordered list.

• First the contents of the @doc array is loaded into a list of scalar variables.

• For each doctor found, the doctor’s <li> list item contains the following HTML sequence:

  Creating an Internet Database (The Result)
  – If the doctor has a Web site, the <a href=$site> opening link tag is written using the site URL from the database record; otherwise, the <b> bold tag is written.
  – The doctor’s full name is written.
  – If the doctor has a website, the </a> closing tag is written; otherwise, the closing bold tag </b>.
  – The doctor’s specialty, title if not blank, address, and second address line if not blank, city, zip, phone number in italics, and languages if not blank, are written.

Maintaining an Internet Database

• There are three tasks involved in database update:
  – Entering new data for the record
  – Making the requested update at the correct position in the file
  – Creating the new version of the file

• Figure 10.8 illustrates the update page.

• The first and last names are required fields.
• The user select the add, change, or delete box and press the “Do it!” button to execute the CGI script.
• Figure 10.9 illustrates the Perl script in an appearance of structured programming.
• First subroutine does CGI interpretation. Second subroutine generates the basic HTML header.
• The &finddoc subroutine reads through file and locate record to modify.

Maintaining an Internet Database
• As long as the user entered a first and last name, the &finddoc subroutine is executed.
• If no doctor is found, &genfounderr will generate some error message.
• &finddoc first opens the existing date file for input and a new data file for output.
• If the record to be updated has not yet been found, look for it; otherwise, just write rest of the file out.

Maintaining an Internet Database
while ($buffer = <DRS>)
  chomp $buffer;
  if (!$found) {
    find a doctor
  } else {
    print NEWDRS "$buffer\n";
  }
}

Maintaining an Internet Database (Requesting Update)
• We only want to change or delete only if we find a matching record; otherwise, we pass the current record to the new file unchanged.
• We pass the current record to the new file unchanged regardless of whether we add or not, and we only add if we don’t find a
match.

Maintaining an Internet Database (Requesting Update)

- &testdocdel simply omits writing the record to the new file, and displays a DELETED! Message.
- &testdocchange routine tests each CGI field for definition and content, and any field that is not blank replaces its corresponding field in the found record.

Maintaining an Internet Database (Requesting Update)

- The change routine creates a new record using the using the join function to reassemble the @doc array into a single string separated by the : symbol.
- Finally, &testdocchange prints the record to the new file, and splits apart again.
- The hardest thing about the add routine is finding where to add the record.

Maintaining an Internet Database (Requesting Update)

- We want to read through the file and find a record where the last name is smaller than the last name in the record to be added, or the last name is the same and the first name is smaller than the first name in the record to be added.
- The join function used to string the CGI variables together and the new record is written to the file.

Maintaining an Internet Database (Requesting Update)

- The fail-safe subroutines are used to show the informative message before the result is displayed.
- The update result looks very much like the original search page because they use the same code.
Text-Based Internet Database
• Search engines cannot do a real-time search through millions of pages to retrieve an up-to-the-second result.
• They use highly sophisticated relational databases to store word content against URL entries.
• The Perl script will search through several dozen HTML pages, returning a user-friendly list what was found in just a few seconds.

CS 898N – Advanced World Wide Web Technologies
Lecture 14: JavaScript
Chin-Chih Chang
chang@cs.twsu.edu

What is JavaScript?
• JavaScript is an interpreted programming or script language from Netscape.
• It is somewhat similar in capability to Microsoft's Visual Basic, Sun's Tcl, the UNIX-derived Perl, and IBM's REX.
• In general, script languages are easier and faster to code in than the more structured and compiled languages such as Java and C++.

What is JavaScript?
• Script languages generally take longer to process than compiled languages, but are very useful for shorter programs.
• JavaScript is used in Web site development to do such things as:
  – Automatically change a formatted date on a Web page.
  – Cause a linked-to page to appear in a popup window.
  – Cause text or a graphic image to change during a mouse rollover.

What is JavaScript?
• JavaScript uses some of the same ideas found in Java, the compiled object-oriented programming derived from C++.
• JavaScript code can be imbedded in HTML pages and interpreted by the Web browser (or client).
• JavaScript can also be run at the server as in Microsoft's Active
What is JavaScript?

- JavaScript uses some of the same ideas found in Java, the compiled object-oriented programming derived from C++.
- JavaScript code can be imbedded in HTML pages and interpreted by the Web browser (or client).
- JavaScript can also be run at the server as in Microsoft’s Active Server Pages before the page is sent to the requestor.

Differences between Java and JavaScript

- The basic differences between Java and JavaScript are:
  - JavaScript is stored on the host machine as source text. Java is stored on the host machine as compiled bytecode.
  - JavaScript is compiled and run as the page is loaded by the browser. Java is compiled from interpretive bytecode to client native code through the Java Virtual Machine and run only after the applet has fully loaded.
  - JavaScript has limited ability to store cookies on the client machine. Java is restricted from any access to the client file system.
  - JavaScript can control a Java applet. A Java applet cannot control JavaScript.
  - JavaScript running on the server is called server-side JavaScript. Java programs running on the server are called servlets.
  - JavaScript running on the client is just plain JavaScript. Java programs running on the client are called applets.

Differences between Java and JavaScript

- JavaScript can interact with the DOM to make HTML dynamic. Java is restricted to running inside its own sandbox.

As far as Web builders are concerned, the primary functional difference between these two languages is that JavaScript can act within the browser, and dynamically change any element on the Web page.

Java is destined to remain a restricted application.

Data Types

- Data types can be declared as constants or variables. Constants
are assigned and variables are declared using the **var** keyword.

- JavaScript has three elementary data types: boolean, number, and string. Some of examples are shown as follows:

  - `boolean result = true; var result = false;`
  - `number c=186282; var x = 0;`
  - `String today = “Monday”; var blank = “”;`

### Object Data Types

- JavaScript has two object data types: array and object.
- Elementary data types are elementary because they occupy a single predictable memory location.
- More sophisticated data types require methods to implement them and so are created as objects.
- Objects have properties (data with attributes) and methods (functions).

### Object Data Types

- Object data types are created (or **instantiated**, which means we are creating an instance or occurrence of the object) with the **new** keyword.

  ```javascript
  var days = new array ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday");
  var months = new array(12);
  Channel[13] = "UPN";
  
  var name = new array(3);
  name[“first”] = “johnnie”;
  name[“middle”] = “b”
  name[“last”] = “goode”
  ```

- This gives us the flexible way to reference each element.

### Object Data Types

- Arrays elements start at 0, so a 12-element array would have elements 0 through 11.
- Arrays can also be created as a property list.

  ```javascript
  name = new array(3);
  name[“first”] = “johnnie”;
  name[“middle”] = “b”
  name[“last”] = “goode”
  ```

- This gives us the flexible way to reference each element.
name[0] = “johnnie”
name.first=“johnnie”

• There are two reasons to create objects:
  – Creating an object from scratch that can then be assigned properties.
    This would most likely be done to create a database type of structure.
  – To make a new reference to an existing object that has been passed by a
    function call.

Operations
• An **operator** is a symbol that describes a mathematical operation
  to be performed on one or more variables.
• JavaScript supports the usual operators:
  – **Arithmetic operators**: plus (+), minus (-), multiply (*), divide (/), and
    remainder (%).
  – **Comparison operators**: equals (==), not equals (!=), less than (<),
    greater than (>), less than or equal to (<=), greater than or equal to
    (>=).

Operations and Evaluations
  – **Logical operators**: and (&&), or (||).
  – **Assignment operators**: equals (=) and the combination arithmetic
    operators.
• Operators are used to create expressions which, after evaluation,
  assigned to variables or used to test conditions.
• The addition operator can also be used with strings to
  **concatenate** them.
• Conditional expressions are evaluated to a single **true** or **false**
  value.

Statements
• There are three types of statements in JavaScript: if statements,
  loop statements, and conditional statements.
• If statements have the format:
  if (condition) statement; else statement;
• Loop statements are written within the context of these
  conditions:
  – A starting, or initial, condition.
– A condition that changes before or after the loop is executed.
– An evaluation of conditions as to whether the loop should be executed a first time or a next time.
– A section of one or more statements that is repeated.
• Loop statements has the following formats:

**Statements**

for (initial condition statements; repeat condition test; condition change statements) {statements}
while (repeat condition test) {statement}
do {statements} while (repeat condition test)
• break and continue can go inside loops that modify the execution sequence.

**Functions**

• The function definition is a statement which describes the function: its name, any values (known as "arguments") which it accepts incoming, and the statements of which the function is comprised.

function funcName (argument1,argument2,etc)
{ statements; }  

**Functions**

• Generally it's best to define the functions for a page in the HEAD portion of a document. Since the HEAD is loaded first, this guarantees that functions are loaded before the user has a chance to do anything that might call a function.
• Consider, for example, a simple function which outputs an argument to the Web page, as a bold and blinking message:

**Functions**

function boldblink(message) {
document.write("<blink><strong>"+message+"</strong></blink>" ); }

• Some functions may return a value to the calling expression.
• The following function accepts two arguments, x and y, and returns the result of x raised to the y power:

**Functions**

function raiseP(x,y) {

}
total=1; for (j=0; j<y; j++)
{ total*=x; }
return total;  //result of x raised to y power
}

- You call a function simply by specifying its name followed by a parenthetical list of arguments.

Structure

- JavaScript has specific rules that apply to its structure:
  - JavaScript must always occur between `<script>` and `</script>` tags.
  - JavaScript functions and blocks of statements are enclosed in curly braces `{}`.
  - JavaScript evaluation expressions are always enclosed in parentheses `()`.
  - JavaScript array elements are indicated using square brackets `[]`.

Structure

- There are two types of comments, single-line start with `//`. Multi-lines start with `/*` and end with `*/`.
- HTML comments `<!–` and `-->` are not recognized by JavaScript, and in fact are used to hide JavaScript from nonscript-handling browsers, but not from script-savvy browsers.
- JavaScript statements can end with a semicolon.

Writing JavaScript

- One of the first things you have to do provide cross-browser compatibility is to determine what browser the user is running so you can avoid script error messages, browser crashes, and bad-looking pages.
- Both Microsoft and Netscape browsers support JavaScript, but sometimes in slightly different ways.

CS 898N – Advanced World Wide Web Technologies

Lecture 15: Dynamic HTML

Chin-Chih Chang

chang@cs.twsu.edu

Dynamic HTML

- Dynamic HTML is a collective term for a combination of new
Hypertext Markup Language (HTML) tags and options, that will let you create Web pages more animated and more responsive to user interaction than previous versions of HTML.

- Much of dynamic HTML is specified in HTML 4.0.

**Dynamic HTML**

- Simple examples of dynamic HTML pages would include:
  - having the color of a text heading change when a user passes a mouse over it or
  - allowing a user to "drag and drop" an image to another place on a Web page.
- Dynamic HTML can allow Web documents to look and act like desktop applications or multimedia productions.

**Dynamic HTML**

- The features that constitute dynamic HTML are included in Netscape's Navigator and by Microsoft's Internet Explorer.
- While HTML 4.0 is supported by both Netscape and Microsoft browsers, some additional capabilities are supported by only one of the browsers.
- Document Object Model (DOM) defines how HTML objects are exposed to the scripting language

**The Concepts and Features in Dynamic HTML**

- Dynamic HTML is a combination of HTML, style sheets, and a scripting language under the umbrella of DOM.
- Both Netscape and Microsoft support:
  - An object-oriented view of a Web page and its elements
  - Cascading style sheets and the layering of content
  - Programming that can address all or most page elements
  - Dynamic fonts

**Style Sheets**

- A term extended from print publishing to online media, a style sheet is a definition of a document's appearance in terms of such elements as:
  - The default typeface, size, and color for headings and body text
  - How front matter (preface, figure list, title page, and so forth) should look
Style Sheets
– How all or individual sections should be laid out in terms of space (for example, two newspaper columns, one column with headings having hanging heads, and so forth).
– Line spacing, margin widths on all sides, spacing between headings, and so forth
– How many heading levels should be included in any automatically generated Table of Contents
– Any reusable content that is to be included on certain pages (for example, copyright statements)

Style Sheets
• Typically, a style sheet is specified at the beginning of an electronic document, either by embedding it or linking to it. This style sheet applies to the entire document.
• As necessary, specific elements of the overall style sheet can be overridden by special coding that applies to a given section of the document.

Style Sheets
• For Web pages, a style sheet performs a similar function, allowing the designer to ensure an underlying consistency across a site's pages.
• The style elements can be specified once for the entire document by either imbedding the style rules in the document heading or cross-referring (linking to or importing) a separate style sheet.

Cascading Style Sheets
• In the case of cascading style sheets, the cascade involves multiple sets of style tags set up in a succession of stages accumulates from one to the next.
• The term cascading refers to the hierarchy of style attributes that are applied to an HTML tag.
• This provides the designer the advantage of being able to rely on the basic style sheet when desired and overriding it when desired.

Cascading Style Sheets
• The filling in or overriding can occur on a succession of "cascading" levels of style sheets.
• One style sheet could be created and linked to from every Web page of a Web site as the overall style sheet.
• For any portion of a page that included a certain kind of content such as a catalog of products, another style sheet that amends the basic style sheet could be linked to.

Cascading Style Sheets

• And within the span of a style sheet, yet another style sheet could be specified as applying to a particular type of product display.
• When creating Web pages, the use of style sheets is now recommended by the World Wide Web Consortium.

Cascading Style Sheets

• The latest version of the Hypertext Markup Language, HTML 4.0, while continuing to support older tags, indicates which ones should be replaced by the use of style sheet specifications.
• The Web's Cascading Style Sheets, level 1 (CSSL1) is a recommendation for cascading style sheets that has been developed by a working group of the World Wide Web Consortium (W3C).

Cascading Style Sheets

• CSS gives more control over the appearance of a Web page to the page creator than to the browser designer or the viewer.
• With CSS, the sources of style definition for a given document element are in this order of precedence: inline styles, embedded styles sheet, and linked style sheet.

Cascading Style Sheets

• Style definitions can actually be placed in three locations: inline styles, embedded styles sheet, and linked style sheet.
• Inline styles can be applied to individual tags in the body section of the page by using the style = attribute within the tags themselves.
Most HTML tags now accept this attribute.

For example:

Cascading Style Sheets

```html
<p style="font-size:18pt; font-style:Arial, Helvetica">Designed by</p>
```

An embedded style sheet is a set of styles enclosed by a set of style tags. For example:

```html
<style>style sheet attributes</style>
```

A linked style sheet can enclose a style list in a separate style sheet file which we link to in the head section. For example:

```html
<link href="pagestyle.css" rel=stylesheet>
```

Cascading Style Sheets

You can have all three types of style sheet markup in the same document. The linked style sheet can be used to declare a base format for an entire Web site, the embedded style sheets can override certain styles in the individual page, and the inline styles have the last word.

Using JavaScript to Make HTML Dynamic

JavaScript brings the capability to write an interactive program to HTML. This is done by applying the features of the JavaScript language to the content of the HTML document.

This comes in the form of dynamic positioning, dynamic content, and events.

**Dynamic positioning** allows you to tell the browser exactly where to put a block of content without using tables.

Using JavaScript to Make HTML Dynamic

**Dynamic content** lets you take a single block of content anywhere in a page and link an event to JavaScript that can update, replace, or remodel it at any time.

When we’re running JavaScript in a Web browser, we receive information on what the user is doing with the mouse and keyboard. This is called **monitoring events**.

Using JavaScript to Make HTML Dynamic
• Some useful events are: onmousemove, onmouseover, onmouseout, onclick, and onchange.
• Some functions we can perform are:
  – Calculating the total amount of an order and displaying the results for the buyer’s approval.
  – Changing the display characteristics of elements defined in a style sheet.
  – Allowing the user to move things around on the page.

Using JavaScript to Make HTML Dynamic
  – Moving elements around on the page without asking the user.
  – Triggering changes on page content based on a timer.
• Events are linked together through what is called the Document Object Model (DOM).
• The DOM was originally created by Netscape for the purpose of using JavaScript, also invented by Netscape.

Using JavaScript to Make HTML Dynamic
• To cause an event to trigger a JavaScript function to access a CSS element, the following steps need to occur:
  – A style sheet is written
  – The target HTML element is given a name attribute.
  – The activating HTML element is given an event attribute that calls a JavaScript function.

Using JavaScript to Make HTML Dynamic
  – The JavaScript function is written to modify the DOM element with the name attribute.
• Embedded style sheets begin with the <style> tag. The type attribute should be declared but the default type is “text/css”, giving us <style type = “text/css”>

DOM

• Document Object Model (DOM) binds JavaScript to HTML, XML, and images in a Web page.
• This means that we are developing a model in which the document or Web page contains objects (elements, links, etc.) that can be manipulated.
DOM

• So you will be able to delete, add, or change an element (as long as the document is still valid, of course!), change its content or add, delete or change an attribute.
• The DOM API provides a standardized, versatile view of a document's contents.

DOM

• By supporting the DOM API, a program not only allows its data to be manipulated by other routines, but does so in a way that allows those manipulations to be reused with other DOMs, or to take advantage of solutions already written for those DOMs.
• The intent is that -- if you stick with the standardized APIs -- any DOM implementation can be plugged together with any DOM-based application.

DOM

• The intent is that -- if you stick with the standardized APIs -- any DOM implementation can be plugged together with any DOM-based application.
• The original example of this was dynamic-HTML scripts; by agreeing on the DOM as their standard representation of the document, scripts can be written that will work properly on all browsers.

DOM

• But this applies to larger-scale programming as well; for example, a server-side solution might be built out of the following reusable components, which may or may not all share a single DOM implementation:
  – A database which presents its contents as a DOM tree.
  – An XML parser which generates a DOM tree, used to read a style sheet.

  – An XSLT processor which combines these, producing a new DOM tree.
  – A routine which writes a DOM's contents out to the network in the
desired syntax (XML, HTML, or other).

- If a better implementation of one of these modules becomes available (a faster XML parser, for example), you should be able to unplug the existing connections and plug in the new component with minimal recoding.

**DOM**

- The DOM Level 1 and Level 2 specifications are W3C Recommendations. This means that the specification is final and can be implemented without fear of things changing.
- Level 1 allows navigation around an HTML or XML document, and manipulation of the content in that document.
- Level 2 extends Level 1 with a number of features: XML Namespace support, filtered views, ranges, events, etc.

**DOM**

- A DOM implementation (also called a host implementation) is that piece of software which takes the parsed XML or HTML document and makes it available for processing via the DOM interfaces.
  - A DOM application (also called a client application) is that piece of software which takes the document made available by the implementation, and does something to it.

**DOM**

- A script which runs in a browser is an example of an application.
- Your favorite browser might implement a JavaScript or VBScript interface, so you can use those scripting languages within the page itself to manipulate the page or change the CSS style sheet.
- Your favorite editor might implement a Scheme or Java interface so you can write an executable in those languages that talks to your editor to manipulate the page.

**DOM**

- The Object Management Group Interface Definition Language (OMG IDL) was chosen as it was designed for specifying language and implementation-neutral interfaces.
- It is expected that the DOM can be implemented using CORBA, COM, or Java Virtual Machine runtime bindings.
DOM

- We expect that many implementations of the DOM will use bindings to various programming languages.
- The DOM specifies bindings for Java and ECMAScript (the standardization of JavaScript/Jscript);
- Other language bindings (for example, ANSI C++, Perl, or VBScript) may be supplied by other interested parties.

CSS Style Attributes

- The common useful CSS attributes are fonts, backgrounds, text, and events.
- Fonts has these elements: font family, font size, font style, font weight, font variant, line height, font.
- Backgrounds have these elements: background color, background repeat, background attachment, background position, and background.

CSS Style Attributes

- Texts have these elements: word spacing, letter spacing, text align, and text indent.
- Events has these elements: onload, onfocus, onblur, onchange, onmouseover, onmouseout, onmousedown, onmouseup, onmousemove, onclick, onkeypress, onkeydown, onkeyup, onsubmit, and onrest.