MASTER OF COMPUTER APPLICATIONS

MCA-12

WEB DESIGNING

Directorate of Distance Education
Guru Jambheshwar University of Science & Technology
Hisar - 125001

CONTENTS

SUBJECT: WEB DESIGNING	
COURSE CODE: MCA-12	**AUTHOR: DR. SUDHANSHU GAUR**
LESSON NO. 1	

WEB INFORMATION

STRUCTURE

1.0 LEARNING OBJECTIVE

After going through this unit, you will be able to:

- Gain the concept of Information Architect
- Learn about all the challenges of Organizing Information.
- Learn about various Organization Schemes.
- Learn about different Organization Structures.
- Gain the concept of Navigation System.
- How to design the Elegant Navigation System.

1.1 INTRODUCTION

This chapter define the role of Information Architect and its importance. While defining the information architecture, we have to face various challenges. We will have a look at different challenges. While defining a website, we have to deal with different organization schemes and different organization structure, we will define all of these. With the combination of organization schemes and structures, we will try to build a cohesive organization system.

Navigation System is the heart of a website. We will understand the importance of Navigation System and its role in the website. With the help of navigation system, we will learn to build a strong elegant navigation system.

1.2 Role of Information Architect

Information Architect is a person who creates the structure or map of information which allows other to find their personal paths to knowledge.

1.2.1 Need of Information Architect

Each building serves its purpose uniquely. Architecture and design of a building depends upon the purpose, location, users, finance etc. if we start constructing a building without deciding its design and architecture the constructors will have problems in constructing it, users will have problems in using it and the purpose for which the building was constructed will never be achieved. Similarly, websites are resource of information. Each website serves its purpose uniquely. If website is developed without any planning about design and architecture then developer may have problems in organizing the information and maintaining it, users may have problems in using the website in searching and accessing the information. These problems may be like time consuming search, time wastage in loading of web page due to improper formats used and difficulty in browsing due to the use of improper keywords.

So information architecture is necessary:

1).For producer so that any updation in the information can be done efficiently within time.

2).For any website to be commercially successful because if user are facing difficulty in searching and navigating the information then they will not use the website again.

3).Because unorganized information can't be converted into knowledge.

Main Job of information Architect

The main jobs if the information architects are given below. An Information Architect

1). Clarifies the mission and vision for the site, balancing the needs of its sponsoring organization & the needs of its audiences.

2). Determines what content and functionality the site will contain.

3).Specifies how users will find information in the site by defining its organization, navigation labeling and searching systems.

4).Maps out how the site will find accommodate change and growth over time.

The Consumer's perspective

Users want to find information quickly and easily. Poor Information Architecture makes busy users confused, frustrated and angry. Because different users have varying needs it is important to support multiple modes of finding information. From the consumer's perspective there can be two modes of finding information.

Known item searching: Some users know exactly what they are looking for. They know what it is called and know it exists. This is called known item searching.
Casual Browsing: Some users don't know what they are looking for. They don’t know the right label. They casually browse or explore the site and they may learn that they have never even considered.
If you care about the consumer, make sure that your information architecture supports both modes. While attractive graphics and reliable technologies are essential to user satisfaction, they are not enough.

The producer perspective

If you are producing an external website the users can be actual or prospective customers, investors, employees, business partners, media & senior executives. If you are producing an intranet the employees of your organization are the consumers. The cost of designing and implementing the architecture is the cost of time spent:

1). In deciding categories of various users.
2). In arguing over the main areas of content and functionality that the site would include.
3).Redesigning.
4).In maintaining the information space on increase in information.

The role of information Architect is to minimize their cost. If information Architect doesn't take care of producer s perspective the burden will be on the site's user to understand how to use and find information in a confusing, poorly designed website. The site maintainers wouldn't know where to locate the new information that the site would eventually include, they had likely to quarrel over whose content was more important and deserved visibility on the main page and so on.

1.2.2 Types of Information Architect

An insider who can understand the sites sponsoring organization.

Advantages

1). Organization’s information is in safe hands.
2). No extra cost, so cost effective.
3). Insider knows the most about in organization's processes and how to get things done within that organization.

Disadvantages

1). Knowledge of an insider may be too specific.

2). Insider may lack the political base required to mobilize cooperation from others in the organization.

3). Insider gets diverted from his original duties.

Someone who can think as an outsider and be sensitive to needs of site's users.

Advantages

1).No biased behavior is expected from an outsider.

2).We have a choice for outsider so we'll choose according to our needs so he'll act more efficiently than insider because he'll be the specialist of his field.

Disadvantages

1). Extra cost.

2). Outsider doesn't have minute details of 'organization so he needs information.

3). Passing secret information of the organization to an outsider can be dangerous.

Outsider can be from a variety of fields like:

Journalism: Journalists are good at editing and organizing information. They have rich knowledge base.

Graphic Design: Graphic Design is much more than creating pretty pictures. It is geared more towards creating relationship between visual elements and determining their effective integration as a whole.

Information and library science: People from this background are good to work with searching, browsing, and indexing technologies.

Marketing: Marketing specialists are expert at understanding audiences and communicating a message effectively to different audiences. They know how to highlight a positive feature and how to suppress the negative ones.

Computer science: Programmers and computer specialists bring an important skill to information architecture. Especially to architecting information from the bottom up. For example, often a site requires a data base to serve the content; this minimizes maintenance and data integrity problems. Computer scientists have the best skills for modeling content for inclusion in a database.

1.2.3 Balance Your Perspective

Whomever you do use as an information architect remembers everyone (including us) is biased by their disciplinary perspective. If possible, try to ensure that other disciplines are represented on your web site development team to guarantee a balanced architecture.

Also, no matter your perspective the information architect ideally should be solely responsible for the site's architecture and not for its other aspects. It can be distracting to be responsible for other more tangible aspects if the site, such as its graphic identity. In this case the site's architecture can easily, it unintentionally, gets relegated to secondary status because the architect is concentrating, naturally on the tangible stuff.

1.3 Collaboration & Communication

The information architect must communicate effectively with the web site development team. This is challenging since information architecture is highly abstract and intangible. Besides communicating the architecture verbally documents (such as blueprint diagrams) must be created in ways that can be understood by the rest of the team regardless of their own disciplinary backgrounds.

Need of Team

In the early days of the web, web sites were often designed, built and managed by a jingle individual through sheer force of will. This webmaster was responsible for amble g and organizing the content, designing the graphics and hacking together any necessary CGI scripts. The only prerequisites were a familiarity with HTML and a willingness to learn on the job. People with an amazing diversity of backgrounds suddenly became webmasters overnight, and soon found themselves torn in many directions at once. One minute they were information architects, then graphics designers, then editors, then programmers.

Then companies began to demand more of their sites and consequently of their webmasters. People wanted more content better organization, greater function and prettier graphics, tables, VRML, frames. Shockwave, Java, and Actives were added to the toolbox. No mortal Webmaster could keep with the rising expectations and increasing complexity of the environment. Increasingly, webmasters and their employers began to realize that the successful design and production of complex web sites requires an interdisciplinary team approach. The composition of this team will vary, depending upon the needs of a particular

project, available budget and the availability of expertise. However, most projects will require expertise in marketing, information architecture, graphic design, programming and project management.

Marketing

The marketing team focuses on the intended purposes and audiences for the web site. They must understand what will bring the right people lo the web site and what will bring them back again.

1.4 Organization Information

We organize information

- To understand
- To explain
- To control

As information architects we organize interrelation so that people can find the right answers to their question.

Organizing Information involves three steps

Structuring

Structuring information means determining appropriate levels of granularity for information atoms in your site and deciding how to relate them to one another.

Grouping

Grouping information means grouping the linked information atoms into meaningful and distinctive categories.

Labeling

Labeling information means figuring out what to call these categories.

Among these structuring and grouping are considered mainly as organizing labeling is done as a separate step.

1.5 Challenges of Organizing Information

High rate of growth of information

The world producers between 1 and 2 exabyte of unique information per year. Given that an exabyte is a billion gigabytes (were talking 18 zeros) this growing mountain of information should keep us all busy for a while. In an unorganized information space any addition of information requires a lot of time to search for the point where to insert the new information atoms.

Ambiguity

Organization systems are built upon the foundation language and language is ambiguous words are capable of being understood more than one way. Think about the word pitch when I say pitch what do you hear? There are more than 15 definitions including:

- A throw, fling or toss.
- A black, sticky sabotage: used for waterproofing.
- The rising and falling of the bow and stern of a ship in a rough sea.
- A salesman's persuasive line of talk.
- An element of sound determined by the frequency of vibration.

This ambiguity results in a shaky foundation for our organization systems. When we use words as labels for our categories we run the risk that users will miss our meaning. Not only do we need to agree on the labels and their definitions, we also need to agree on which documents to place in which categories. Consider the common tomato. According to Webster's dictionary a tomato is a red or yellowish fruit with a juicy pulp used a vegetable but botanically it is a berry if we have such problems classifying the common tomato considers the challenges involved in classifying web site content.

Heterogeneity

Heterogeneity refers to an object or collection of objects composed of unrelated or unlike parts. At the other end of the scale homogenous refers to something composed of similar or identical elements. An old fashion library card catalog is relatively homogenous. It organizes and provides access to books. It doesn't provide access to chapters in books or collection of books. It may not provide access to magazines or videos. This homogeneity

allows for a structured classification system. Each book has a record in the catalog. Each record contains the same fields like author, title and subject.

Most websites on the other hand are highly heterogeneous in many respects. For example, websites often provide access to documents and their components at varying levels of granularity. A website might present articles and journals and journal database side by side. Links might lead to pages, section of pages or other web sites. Also, websites provide access to document in multiple formats; you might find financial news, product description, employee home page, image gallery and software files. Dynamic news contains share apace with static human resources information shares with videos, audios and interactive applications. The heterogeneous nature of website makes it difficult to impose any single structured organization system on the content.

Difference in Perspective

The fact is that labeling and organization systems are intensely affected by their creators perspective. we see this at the corporate level with websites organized according to internal division organization charts with grouping such as marketing, sales, customer support, human resources and information systems. How does a customer visiting this web site know where to go for technical information about a product they just purchased? To design usable systems, we need to escape from our own mental models of content labeling and organization.

Internal Politics

Politics exist in every organization, individuals and departments constantly position for influence or respect. Because of the inherent power of information organization in forming understanding and opinion, the process of designing information for website and internets can involve a strong undercurrent of politics. The choice of organization and labeling systems can have a big impact on how users of sites perceive the company, its departments and its products. For example should we include a link to library site on the main page of the corporate internees? Should we call it the library organization information services or knowledge management? Should information resources provided by other departments be included in this area? If the library gets a link on the main page, then why not corporate communications? What about daily news? As an information architect, you must be

sensitive to your organization's political environment. Politics raise the complexity and difficulty of creating usable information architectures. However, if you are sensitive to the political issues at hand, you can manage their impact upon the architecture.

1.6 Organizing Website and Internet

The organization of information websites is a major factor information determining success. Organization systems are composed of organization schemes and organization structure.

An organization schemes defines the shared characteristics of content items and influence the logical grouping of those items. An organization structure defines the types of relationships between content items and groups.

1.6.1 Organization schemes

Various organization schemes are being used today. These schemes can be divided into two categories:

1). Exact organization schemes

2). Ambiguous organization schemes

1.6.1.1 Exact organization schemes

These schemes divide the information into well-defined and mutually exclusive sections. Users can search the information only if he knows what he is looking for and he knows the label so that he can identify the group/section in which the item is. This is known as well-defined and mutually exclusive known item searching. No ambiguity is involved.

Advantages:

1).Exact organization schemes are easy to design and maintain because there is little intellectual work involved in assigning items to categories.

2).Use.

Disadvantages:

1).Exact organization schemes require the user to know the specific name of the resource they are looking for.

Examples

Alphabetical organization scheme

In this scheme all the information atoms are arranged alphabetically and they are grouped accordingly i.e. atoms starting with letter ‘A’ come information one group and so on. The implementation of this scheme can be observed information encyclopedias, Dictionaries, phone books, bookstores, departmental store directories. On the web you can observe this scheme information the Address book of your mailbox.

Chronological Organization Schemes

Certain types of information lend themselves to 'chronological organization. E.g. an archive of press releases might be organized by the date of release. History books, magazine archive, diaries and television guides tend to be organized chronologically. As long as there is agreement on when a particular event occurred, chronological schemes are easy to design and use.

Geographical Organization Schemes

Place is often an important characteristic of information. Political, social and economic issues are frequently location dependent. We care about the news and weather that affects us information our location. Example, in a website of MNC the list products available is different in different countries according to economy, population of that country. So such a website manages such location dependent information using Geographical Organization Schemes.

1.6.1.2 Ambiguous Organization Schemes

Ambiguous Organization Schemes divide information into categories that defy exact definition. They are mired in the ambiguity of language and organization.

Advantages

Ambiguous Organization Schemes are more important useful than exact organization schemes because the information atoms are grouped on the basis of there meaning not just because they start from a particulars alphabet. This grouping of related items supports associative learning process that may enable the user to make new connections and reach better conclusions.

Disadvantages

Ambiguous Organization Schemes are difficult to

- Design
- Maintain
- Use
-

Examples

Topical Organization scheme

Organizing information by topic or subject is one of the most useful and challenging approaches. In this scheme all the information atoms related to one topic is grouped information a single category E.g. Phone book yellow pages are organized topically. So that is the place to look when you need plumber. Research oriented websites rely heavily on topical organization schemes. In designing a topical Organization Scheme, it is important to define the breadth of coverage. Some schemes such as encyclopedia cover the entire breadth of human knowledge while others such as a corporate website are limited information breadth covering only those topics directly related to the company's products and services. E.g.Yahoo.com categorizes information as Ads & Entertainment, Hobbies and Games, Industry and Business etc. whereas Microsoft.com categorizes as about, Product Support, Careers, and Contacts etc.

Task-oriented Organization Scheme

Task oriented schemes organize content and application into a collection of processes, functions/ tasks. These schemes are appropriate when it is possible to anticipate a limited number of high priority tasks that users will want to perform. E.g. Ms Word uses this scheme as collections of individual actions are organized under task oriented menus such as Edit, Insert and Format. On the web task oriented organization schemes are most common information the context of e-Commerce websites where customer interaction takes center stage.

Audience-specific Organization Scheme

In case where there are two or more clearly definable audiences for a website organization intranet an audience specific organization scheme may make sense. This type of scheme works best, when the site is frequently used by repeated visitors who can bookmark their particular section of the site. It also works well if there is value is customizing the content

for each audience. Ancient: oriented schemes break a site Inc: smaller audience specific mini-sites, thereby allowing for clutter free pages that 'present only the options of internet to that particular audience. E.g. anycollege.com contains different links for students, faculty and management. Audience specific schemes can be open or closed. An open scheme allows members of one audience to access the content intended for other audiences. A closed scheme prevents members from moving between audience specific sections.

Hybrid Organization Scheme

The power of a pure organization scheme derives from its ability to suggest a simple mental model that users can quickly understand. However when you start blending elements of multiple schemes. Confusion often follows and solutions are rarefy scalable. The hybrid scheme can include elements of audience specific, topical task- oriented and alphabetical organization schemes. Because they are all mixed together we can't form a mental model. But as it is often difficult to agree upon any one scheme hybrid schemes are fairly common.

1.6.2 Organization structures

Organization Structure plays an intangible yet very important role in the design websites. The structure of information defines the primary ways in which users can navigate. Major organization structures that apply to web site and internet architectures include

- The Hierarchy
- The database-oriented Model
- Hypertext

Each Organization structure possesses unique strengths and weaknesses.

1.6.2.1 The Hierarchy/taxonomy: A Top-Down Approach

In this model we give the groups a hierarchical structure having parent child relationship between them such that they get divided into mutually exclusive groups.

Examples

- Family trees are hierarchical.
- Organization charts are usually hierarchical.
- We divide books into chapters into sections into paragraphs into sentences into words into letters.

Advantages

1).The mutually exclusive subdivisions and parent child relationships of hierarchies are simple and familiar.

2).Because of this pervasiveness of hierarchy, users can easily and quickly understand web sites that use hierarchical organization models. They are able to develop a mental model of the site's structure and their location within that structure. This provides content that helps users feel comfortable.

3). The top-down approach allows you to quickly get a handle on the scope of the website without going through an extensive content inventory process. You can begin identifying the major content areas and exploring possible organization schemes that will provide access to that content. Because hierarchies provide a simple and familiar way to organize information, they are usually a good place to start the information architecture process.

Designing the modal

While designing the hierarchical model we should take care of the following points.

Balance between exclusivity and exclusivity

The hierarchical categories should be mutually exclusive. Within a single organization scheme there is a need to balance the between exclusivity and exclusivity. Hierarchical model that allows listing is known as polyhierarchical modal. But if too many items are cross listed then hierarchy loses its value.

Balance between breadth and depth

It is important to consider the balance between breadth and depth of the hierarchy. Breadth refers to the number of options at each level in the hierarchy and depth refers to the number of levels at each level in the hierarchy. If a hierarchy is too narrow and deep then users have to click through an inordinate number of levels to find what they are looking for. In the (relatively) broad and shallow hierarchy users must choose from a large number (say ten) of categories and may get unpleasantly surprised by the lack of content once they select an option.

1.6.2.2 The Database Model: A Bottom-up Approach

Metadata is the primary key that links information architecture to the design of database

schema. Metadata allows us to apply the structure and power of e relational databases to the heterogeneous, unstructured environments of web sites and intranets. By tagging documents ally other information objects with controlled vocabulary metadata, we enable powerful searching and browsing. This is a bottom up solution that works well information large distributed environments.

It's not necessary for information architects to become experts in SQL, XML schema definitions, the creation of ER Diagrams and the design of relational databases, though these are all extremely valuable skills. Instead, Information Architects need to understand how metadata, controlled vocabularies, and database structures can be used to enable:

- Automatic generation of alphabetical indexes.
- Dynamic presentation of associative "see also" links.
- Field searching
- Advanced filtering and sorting of search results.

The database model is particularly useful when applied within relatively homogeneous sub sites such as product catalogs and staff directories.

1.6.2.3 Hypertext Model

Hypertext is relatively recent and highly nonlinear way of structuring information. Hypertext system involves two primary types of components: the items organization chunks of information that will be linked and the links between these chunks. These components can form hypermedia systems that connect text, data, Image, video and audio chunk's. Hyper-text chunks can be connected hierarchically, non-hierarchically or both. In hypertext systems, content chunks are connected via links in a loose web of relationships.

Advantages

1).This model provides great flexibility.

2).This model allows for useful and creative relationships between items and areas in the hierarchy. If usually makes sense to the first design the information hierarchy and then identify ways information which hypertext can complement the hierarchy.

Disadvantages

1).This model introduces substantial potential for complexity and confusion because hyper text links reflect highly personal associations.

2).As users navigate through highly hyper textual websites it is easy for them to get lost.

3).Hyper textual links are often personal information nature. The relationship that one

person sees between content items may not be apparent to others.

1.6.3 Creating Cohesive Organization Systems

Organization systems are fairly complex. We have so many options for choosing appropriate organization scheme and appropriate organization structure. Taken together in the context of a large website development project, the choice of a proper system becomes difficult that's why it is important to break down the site into its components, so you can tackle one option for scheme/structure at a time. Also we know that all information retrieval systems work best when applied to narrow domains of homogeneous content we can identify opportunities for highly effective organization systems, However it's also impotent not to lose sight of the big picture.

In considering which organization scheme to use remembers the distinction between exact and ambiguous schemes.

Exact schemes are best for known item searching. Ambiguous schemes are best for browsing and associative learning. Whenever possible use both types of schemes. Also beware of the challenges of organizing information. When thinking about which organization structure to use, keep information mind that large websites and intranets typically require all three tapes of structures. The top-level, umbrella architecture for the site will almost certainly be hierarchical. While designing this hierarchy we should keep a lookout for collection of structures homogeneous information. These potential subsides are excellent candidates for the database model. Finally, less structured and more creative relationships between content items can be handled through typeset. In this way all three organization structures together can create a cohesive organization system.

1.7 Navigation Systems

When we have lards amount of information then to organize the information space we divided the information space into groups and label them. To look for any information atom we need to search for its link information its group. This is done using browsing/navigation user can get lost in the information space but a well-designed taxonomy may reduce the chances that user will become lost. So generally, Navigation Systems are beneficial if information is organized using the hierarchy model. Complementary navigation tools are often needed to provide context and to allow for greater flexibility.

Navigation and Searching

Navigation and Searching both are used for finding information. Navigation searches for the information to be found by moving between links available. But information searching we give the information about the information to be found as text to the search engine and search engine does the task of finding information for users. We can search for a phrase but can't navigate.

1.7.1 Types of Navigation Systems

1.7.1.1 Embedded/integrated Navigation Systems

Embedded Navigation Systems are typically wrapped around and infused within the content of the site. These systems provide context and flexibility helping users understand where they are and where they can go. Embedded Navigation Systems can be further divided into three categories:

Global (site-wide) Navigation System: By definition, a global navigation system is intended to be present on every page throughout a: site. It is often implementing in the form of a navigational bar at the top of each page. These site wide navigation systems allow direct access to key areas and functions, no matter where the user travels in the site's hierarchy. Most global navigation systems provide a link to the home page. Many provide a link to the search function.

Local Navigation Systems: Local Navigation Systems enable users to explore the immediate area. Some lightly controlled sites integrate global and local navigation into a coexistent unified system. A user who selects business sees different nevi anion options than a reader who selects sports, but both sets of options area presented within the same navigation framework. These local navigation systems and the content to which they provide access are often so different that these local areas are referred to as sub sites or sites within sites. Sub sites exist because (1) areas of content and functionality really do merit a unique navigation approach (2) due to decentralized nature of large organization different groups of people are often responsible for different content areas and each group may decide to handle navigation differently.

Contextual Navigation system: Some relationships don't fit neatly into the structured categories of global and local navigation. This demands the creation of contextual navigation links specific to a particular page, document or object. E.g. Words or phrases within sentences are represented as embedded or inline hypertext links. On an e-commerce

site, these “See Also” links can point users to related products and services. In this way contextual navigation supports associative learning. Users learn by exploring the relationship you define between items. They might learn about useful products they didn't know about.

1.7.1.2 Supplemental/ Remote Navigation System

These navigation systems are external to the basic hierarchy of a website and provide complementary ways of finding content and completing tasks. These navigation systems provide users with and emergency backup. Some of the examples of Remote navigation Systems are

Sitemaps: In a book/ magazine, the table of contents presents the top few levels of the information hierarchy. It shows the organization structure for the printed work and supports random as well as linear access to the content through the use of chapter and page numbers. In context of websites a sitemap provides a board view of the content in the website and facilities random access to segmented portions of that content. A sitemap can employ graphical or text based links to provide the user with direct access to pages of the site. A sitemap is the most natural for websites that lend themselves to hierarchical organization. But for a small website with only two or three hierarchical levels a sitemap may be unnecessary.

Site Indexes: Similar to the back of book index found in many print materials, a web based index presents keywords organization phrases alphabetically, without representing the hierarchy. Unlike a table of contents indexes are relatively flat, presenting only one or two levels of depth. Therefore indexes work well for users who already know the name of the item they are looking for. Large complex Websites often require both a sideman and a site index. For small sites, a site index alone may be sufficient. A major challenge in indexing a website involves the level of granularity.

Methods to create index are

1).For small sites create content to inform decisions about which links to include.
2).For large sites, use controlled vocabulary indexing at the document level to drive automatic generation of tie site index.

Guides: Guides take several forms including guided tours, tutorials and micro portal focused around a specific audience topic or task. In each case, guides supplement the existing means of navigating and understanding site content. Guides typically feature linear navigation but hyper textual navigation should be available to provide additional flexibility.

Rules for designing guides:

1).The guide should be short.

2).At any point, the user should be ably to exit the guide.

3).Navigation should be located information the same spot on every page so that users can easily step back and forth through the guide.

4).The guide should be designed to answer questions.

5).Screenshots should be crisp, clear and optimized with enlarged details of key features.

6).If the guide includes more than a few pages, it may need its own table of contents.

Uses of Guides

1).Guides often serve is a useful toil for introducing new users to the content and functionality of a website.

2).Guides can be valuable marketing tools for restricted access websites enabling you to show potential customers what they will get for their moneys.

3).Guides can be valuable internally, providing an opportunity to showcase key features of a redesigned site to colleagues, managers and venture capitalists.

Linking between Navigation and searching: Searching is loosely linked with integrated Navigation Systems and tightly linked with Remote Navigation systems.

1.7.2 Designing Elegant Navigation Systems

Designing navigation systems that work well is challenging. You've got so many possible solutions to consider and lots of sexy technologies such as pop-up menus and dynastic site maps can distract you from what's really important: building context, improving flexibility, and helping the user to find the information they need. No single combination of navigation elements works for all web sites. One size does not fit all. Rather, you need to consider the specific goals, audience, and content for the project at hand, if you are to design the optimal solution.

However there is a process that should guide you through the challenged of navigation system design. It begins with the hierarchy. As the primary navigation System, the hierarchy influences all other decisions. The choice of major categories at the highest levels of the website will determine design of the global navigation system. Based on the hierarchy, you will be able select key pages or types of pages that should be accessible from every other page on the web site in turn, the global navigation system will determine design

of the local and then ad hoc navigation systems. At each level of granularity. You design of the higher order navigation system will influence decision at the next level.

Once you have designed the integrated navigation system, you can consider the addition of on or more remote navigation elements. In most cases, you will need to choose between a table of contents, an index, and a sitemap. Is the hierarchy strong and clear? Then perhaps a table of contents makes sense. Does the hierarchy get in the way? Then you might consider an index. Does the information lend it self to visualization? If so, a sitemap may be appropriate. Is there a need to help new or prospective users to understand what they can do with site? Then you might add a guided tour.

If the site is large and complex, you can employ two or more of these elements. A table of contents and an index can serve different users with varying needs. However, you must consider the potential user confusion caused by multiple options and the additional overhead required to design and maintain these navigation elements. As always, it's a delicate balancing act.

If life on the high wire unnerves you be sure to build some usability testing into the navigation system design process. Only by learning from users can you design and reline an elegant navigation system that really works.

1.8 Check Your Progress

1. Two types of Information Architect are …… and …….
2. From the consumer's perspective there can be …… modes of finding information.
3. Some users don't know what they are looking for, this type of searching is called…….
4. Two types of organization schemes are defined as…… and ……
5. Geographical Organization Schemes is the example of ……. organization scheme.
6. Hybrid Organization Scheme is the example of ……. Organization scheme.
7. Topical Organization Scheme is the example of ……. Organization scheme.

8. Database model is also known as ……. approach.
9. Sitemaps comes under ……. Type of Navigation System.
10. Local Navigation system is a type of……. Type of Navigation System.

1.9 Summary

Information Architect is a person who creates the structure or map of information which allows other to find their personal paths to knowledge.
Information architecture is necessary:
1).For producer so that any updation in the information can be done efficiently within time.
2).For any website to be commercially successful because if user are facing difficulty in searching and navigating the information then they will not use the website again.
3).Because unorganized information can't be converted into knowledge.

The information architect must communicate effectively with the web site development team. This is challenging since information architecture is highly abstract and intangible. Besides communicating the architecture verbally documents (such as blueprint diagrams) must be created in ways that can be understood by the rest of the team regardless of their own disciplinary backgrounds.

Various challenges in organization system are:

- High rate of growth of information
- Ambiguity
- Heterogeneity
- Difference in Perspective
- Internal Politics

The organization of information websites is a major factor information determining success. Organization systems are composed of organization schemes and organization structure. An organization schemes defines the shared characteristics of content items and influence the logical grouping of those items. An organization structure defines the types of relationships between content items and groups.

Various organization schemes are being used today. These schemes can be divided into two categories:

1). Exact organization schemes

2). Ambiguous organization schemes

Organization Structure plays an intangible yet very important role in the design websites. The structure of information defines the primary ways in which users can navigate. Major organization structures that apply to web site and internet architectures include

- The Hierarchy
- The database-oriented Model
- Hypertext

Navigation and Searching both are used for finding information. Navigation searches for the information to be found by moving between links available. But information searching we give the information about the information to be found as text to the search engine and search engine does the task of finding information for users. We can search for a phrase but can't navigate.

Two types of Navigation System are:

1.) Embedded/integrated Navigation Systems

2.) Supplemental/ Remote Navigation System

1.10 Self-Assessment Test

Q.1 Describe the role of Information Architect briefly.

Q.2. What are various challenges in Organization System?

Q.3. Describe the various organization schemes. Explain Briefly.

Q.4. Explain different organization structures.

Q.5. Explain Navigation System and its importance.

Q.6. Explain different types of Navigation System.

1.11 Answers to check your progress

1. Insider and outsider
2. Two
3. Casual Browsing
4. Exact and Ambiguous
5. Exact
6. Ambiguous
7. Ambiguous
8. Bottom-Up
9. Supplemental/ Remote Navigation System
10. Embedded/integrated Navigation Systems

1.12 References / Suggested Readings

1. Thomas A Powell, HTML-The Complete Reference,Tata McGraw Hill.
2. Scott Guelich, Shishir Gundavaram, Gunther Birzniek; CGI Programming with Perl 2/e. O'Reilly.
3. Doug Tidwell, James Snell, Pavel Kulchenko; Programming Web Services with SOAP, O'Reilly.
4. Pardi, XML in Action, Web Technology, PHI.
5. Yong, XML Step by Step, PHI.
6. Aaron Weiss, Rebecca Taply, Kim Daniels, Stuven Mulder, Jeff Kaneshki, Web Authoring Desk Reference, Techmedia Publications.
7. HTML The complete Reference, TMH

SUBJECT: WEB DESIGNING	
COURSE CODE: MCA-12	**AUTHOR: DR. SUDHANSHU GAUR**
LESSON NO. 2	

SEARCHING SYSTEM

STRUCTURE

2.0 LEARNING OBJECTIVE

After going through this unit, you will be able to:

- Learn about Searching System
- Learn about types of Searching System
- Learn about conceptual design
- Learn about design sketches.
- Learn about different phases of Website Development.

2.1 INTRODUCTION

This chapter define the searching system and its importance in website. We will learn various types of Searching System. We also have a look at the way to design searching system.

We will have a look at conceptual design and blueprints. With the help of Blueprints, we will be able to create design sketches. We will explain the various phases of Website Development.

2.2 Searching Systems

Need of searching systems

1). As the amount of information on the website increases it become difficult to find the required information. If the navigation systems are not properly designed and maintained then to find the required information searching systems are required.

2). If your site has enough contents and users come to your site to look for information then site need searching systems.

3). Search system should be there on your site if it contains highly dynamic contents e.g. web based newspaper.

4).A search system could help by automatically indexing the contents of a site once or many times per day. Automating this process ensures that users have quality access to your website's contents.

Searching your website

Assuming you have decided to implement a Searching system for your website. It's important to understand how users really search before designing it.

Users have different kinds of information need: Information scientists and librarians have been studying user's information finding habits decades. Many studies indicated that users of information systems are not members of a single-minded monolithic audience who want the same kind of information delivered information the same ways. Some want just a little while other wants detailed assessment of everything there is to know about the topic. Some want only the accurate, highest quality information; while others do not care much about the reliability of source. Some will wait for results while others need the information yesterday. Some are just plan happy to get any information at all, regardless of how much relevant stuff are really missing. Users needs and expectation vary widely and so the information systems that them must recognize, distinguish and accommodate these different needs.

To illustrate let's look at one of these factors in greater detail: The variability information users searching expectations.

Known item searching

Some users information needs are clearly defined and have a single correct answer. When you check the newspaper to see how your stock information amalgamated shoelace and aglet is 'doing (especially since the hostile Microsoft takeover attempts), you know exactly what you want that the information exists and where it can be found.

Existence searching

However some users know what they want but do not know how to describe it or weather the answer exists at all e.g., you must want to buy shares information Moldovan high start-ups and that carries no load. You are convinced that this sector is up and coming, but do fidelity and Merrill lynch know this as well'. You might check their Webster, call a broker or two, or ask your in the know aunt. Rather then a clear question for which a right answer exists, you have an abstract idea on concept, and you don't know whether matching information exists. The success of yours search depends as much upon the abilities of the

brokers, the websites, and your aunt to understand your idea and its contexts as whether the information (information in this case a particular mutual fund) exists.

Exploratory searching

Some users know how to phrase their question, but don't know exactly what they are hoping to find and are really just exploring and trying to learn more. lf you ever considered changing careers you know what we mean you are not sure that you definitely what to switch to chinchilla farming, but you have heard it is the place to be, so you might informally ask a friend of a friend who an uncle in the business. Or you call the public library to see if there's a book on the subject, or you write to the chinchilla professionals association requesting more information. In any case, you are not sure exactly what you will uncover, but you are re willing to take the time to learn more. Like existence searching, you have so much a question seeking answer as much as an idea that you want to learn more about.

Comprehensive Searching (Research)

Some users want everything available on a given topic. Scientific researchers, patent lawyers, doctoral students trying to find unique and original dissertation topics, and fans of any sort fit in to this category. For example if you idolize that late great music duo Milli Vanilli, you'll want to see everything that has anything to do with them Single and records, bootlegs, concert tour plasters, music videos, fan club information, paraphernalia, interviews, books, scholarly articles, and records burning schedules. Even casual mentions of the band, such as someone's incoherent ramblings information a web page or Usenet newsgroup, are fair game if you're seeking all there is to know about Milli Vanilla so you might turn to all sorts of information sources for help friends, the library, books stores, music stores, radio call in shows and so on There are many other ways of classifying information needs. But the important thing remember is that not all users are looking for the same thing. Ideally, you should anticipate the most common types of needs that your site's users will have and ensure that these needs are met minimally; you should give some thoughts to the variations and try to design a search interface that is flexible in responding to them.

2.3 Designing the Search Interface

Concept of Searching system

There are two models of searching systems:

1). In the first and older model user express their information need as query that they enter in a search interface. They may do so using a specialized search language.
2). In the second model users express the information need information the natural language like English.

After this step Queries are matched against an index that represent the site's content and a set of matching documents is identified.

Designing the Search Interface

With so much variation among users to account for, there can be no single ideal search interface. Following factors affect choice of search interface:

The levels of searching expertise users have: Are they comfortable with Boolean operators. Or do they prefer natural language? Do they need simple or high powered interface? What about a help page?

The kind of information the user wants: Do they want just a taste or are they doing comprehensive research? Should the results be brief, or should they provide extensive detail for each document?

The type of information being searched is it made up of structured fields or full texts? Is it navigation pages, destination pages, or both? HTML or other formats?

How much information is being searched: will users be overwhelmed by the number of documents retrieved?

Support Different Modes of Searching

Use the same interface to allow users to search the product catalog, or the staff directory, or other content areas. Are non-English speakers important to your site? Then provide them with search interfaces in their native languages. Including language specific directions, search commands and operators, and help information. Does your site need to satisfy users with different levels of sophistication with online searching? Then consider making available both a basic search interface and an advanced one.

Simple / Basic search interface

A simple search interface was required; because at limes users wouldn't need all the firepower of an advanced search interface. Especially when conducting simple known item searches. A simple search box is ideal for the novice or for a user with a pretty good sense of what he or she is looking for. Mammal filtering options are provided including searching for keywords within little and abstract fields, searching within the author field or searching within the publication number field. These filtering options provide the user with more power by allowing more specific searching. But because the labels keyword, Author, And publication Number are fairly self-explanatory. They don't force the user to think too much about these options.

Advanced search Interface

We needed interface that would accommodate this important expert audience who were used to complex Boolean and proximity operators and who where already very used to the arcane search languages of other commercial information services. This interface supports the following types of searching:

Fielded Searching

Author, keyword, Title, Subject and ten other fields are reachable. A researcher could, for example find a dissertation related to his or her area of interest by searching the subject field, and learn who that doctoral student's advisor was by reading the abstract. To find other related dissertations, the researcher could then search the advisor field to learn about other doctoral students who shared the same advisor.

Familiar Query Language

Because many different query language conventions are supported by traditional on line products, users may be used to an established convention. The effort to support these users is made by allowing variant terms. For the field Degree Date the user can enter either ‘‘ddt’’, "da", "date", "Yr " or year.

Longer Queries

More complex queries often require more space than the single line entry box found in the simple search interface. The more complex interface supports a much longer query.

Reusable Result Sets

Many traditional online information products allow searchers to build sets of results that can be reused. In this example, we've ANDed together the two sets that we've already found and could in turn combine this result with other sets during the iterative process of searching. Because this advanced interface supports so many different types of searching we provided a substantial help page to assist users. For users of common browsers, the help page is launched if a separate browser window so that users don't need to exit the search interface to get help.

Searching and browsing systems should be closely integrated

As we mentioned earlier, users typically need to switch back and forth between searching and browsing. In fact users often don't know if they need to search or browse in the first place. Therefore, these respective systems shouldn't live in isolation from one another. The search/browse approach can be extended by making search and browse options available on the search result page as well, especially on null results pages when a user might be at a dead end and needs to be gently led back to the process of iterative searching and browsing before frustration sets in.

Searching should conform to the site's Look and feel

Search engine interfaces and more importantly, retrieval results, should look and behave like the rest of your site.

Search Options Should Be Clear

We all pay lip service to the need for user documentation, but with searching it's really a must Because, so many different variables are involved with searching there are many opportunities for things to go wrong on a help or Documentation page consider letting the user know the following:

What is being searched?

Users often assume that their search query is being run against the full test of every page in your site Instead your site may support fielded searching or another type of selective searching. If they're curious users should be able to find out exactly what they are searching.

How they can formulate search queries

What good is it to build in advanced querying capabilities if the user never knows about them? Shows off the power of your search engine with excellent real life examples. In other words make sure your examples actually work and retrieve relevant documents if the user decides to test them.

User options

Can the user do other neat things much as changing the sorting order of retrieval results? Show them off as well!

What to do if the user can’t find the right information

It is important to provide the user with some tricks to handle the following three situations:

- I'm getting too much stuff
- I'm not getting anything
- I'm getting the wrong stuff

For case (a), you might suggest approaches that narrow the retrieval results. For example if your system suppers the Boolean operator AND, suggest that users combine multiple search terms with an AND between them (ANDing together terms reduces retrieval size).

If they are retrieving zero 'results as in case (b), suggest the operator OR the use of multiple search terms the use of truncation (which wife retrieve a term's use o variants), and so on.

If they are completely dissatisfied with their searches, case(c), you might suggest that they contact someone who knows the site’s content directly for custom assistance, it may be a resource intensive approach, but it’s a far superior last resort to ditching the user without helping them at all.

Choose a search Engine That Fits Users' Needs

At this point, you ideally will know something about the sorts of searching capabilities that your site’s users will require. So select a search engine that satisfies those needs as much as possible for example, if you know that your site’s users already very familiar with a particular way of specifying a query such as the use of operators, then the search engine you choose should also support using Boolean operators. Does the size of your site suggest that users will get huge retrieval results? Be sure that your engine be supports techniques for whittling down retrieval sizes, such as the AND & NOT operators , or that it supports relevance ranked results that list the most relevant results at the top will users have a problem with findings the right terms to use in their search queries?

Display search Results sensibly

You can configure how your search engine displays search results information many ways. How you configure your search engine results depends on two factors.

The first factor is the degree of structure your content has. What will your search engine be able to display besides just the titles of retrieved documents? Is your site's content

sufficiently structured so that the engine can parse out and display such information as an author, a date an abstract, and so on?

The other factor is what your site's users really want. What sorts of information do they need and expect to be provided as they review search results?

When you are configuring the way your search engine displays results you should consider these issues:

1) How much information should be displayed for each retrieved document?

To display less information per result when you anticipate large result sets. This will shorten the length of the results page making it easier to read. To display less information to users who know what they're looking for, and more information to users who aren't sure what they want.

2). What Information should be displayed for each retrieved document?

Which fields you show for each document obviously depends on which fields are available in each document, what your engine displays also depends on how the content is to be used. Users of phone directories for example want phone numbers first and foremost. So it makes sense to show them the information from the phone number field on the results page. Lastly, the amount of space available on a page is limited: You can't have each field displayed, so you should choose carefully and use the space that is available wisely.

3).How many retrieved documents should be displayed?

How many documents are displayed depends on the preceding two factors: If your engine displays a lot of Information for each retrieved document, you'll want to consider a smaller size for the retrieval set, and vice versa. Additionally the user's monitor resolution and browser settings will affects the amount of information that can be displayed individually.

4).How should retrieved document be sorted?

Common options or sorting retrieval results include:

- In chronological order.

- Alphabetically by title, author, or other fiends.
- By an odd thing called relevance.

Certainly, if your site is providing access to press releases or other news- oriented information, sorting by reverse chronological order makes good sense. Chronological order is less common, and can be useful for presenting historical data.

Alphabetical sorts are a good general purpose sorting approach (most users are familiar with the order of the alphabet). Alphabetical sorting works best if initial articles such as a and the are omitted from the sort order (certain search engines provide this option).

Relevance is an interesting concept; when a search engine retrieves 2000 documents, is not it great to have them sorted with the most relevant at the top, and the least relevant at the bottom? Relevance ranking algorithms are typically determined by some combination of the following; how many of the query's terms occur in the retrieved document; how many times terms occur in that document; how close to each other those terms occur and where the terms occur.

Always provide the user with feedback

When a user executes a search, he or she expects result. Usually a query with retrieves at least one document, so the user's expectation is fulfilled. But sometimes a search retrieves zero results. Let the user know by creating a different results page especially for these cases. This page should make it painfully clear that nothing was retrieved, and give an explation as to why, tips for improving retrieval results and links to both the help area and to a new search interface so the user can try again.

Other Considerations

You might also consider including a few easy to implement but very useful things in your engine's search results:

Repeat back the original search query prominently on the results Page

As users browse through search results, they may forget what they searched for in the first place remind them. Also include the query in the page titles; this will make it easier for users to find it in their browser's history lists.

Let the user know how many document in total were retrieved.

Users want to know how many documents have been retrieved before they begin reviewing the results. Let them know: if the number is too large, they should have the option to refine their search.

Let the user know where he or she is in the current retrieved set.

It's helpful to let users know that they're viewing documents 31-40 of the 83 total that they've retrieved.

Always make it easy for the user to revise a search or sort a new One.

Give them these options on every results page and display the current search query on the revise search page so they can modify it without reentering it.

2.3.1 To Search or Not To Search

It's becoming a doubtful question whether to apply a search engine in your site. Users generally expect searching to be available, certainly in large sites. Yet we all know how poorly many search engine actually work. They are easy to set up and easy to forget about. That's why it's important to understand how users information needs can vary so much and to plan and implement your searching system's interface and search zones accordingly.

To find the answer to the question: TO SEARCH OR NOT TO SEARCH?

We need to know the need of Searching and searching Systems, advantages disadvantages of Searching and searching systems. All of these points have already been discussed.....!!!!

2.4 Indexing the Right Stuff

Searching only works well when the stuff that's being searched is the same as the stuff that users want. This means you may not want to index the entire site. We will explain:

Indexing the entire site.

Search engines are frequently used to index an entire site without regarded for the content and how it might vary. Every word of every page, whether it contain real content or help information, advertisement, navigation, menus and so on. However, searching barks much batter when the information space is defined narrowly and contains homogeneous contents. By doing so, the site's architects are ignoring two very important things: that the information in their site isn't all the same. And that it makes good sense to respect the lines already drawn between different types of content. For example, it's cleared that German and English content are vastly different and that there audience's overlap very little (if at all) so why not create separately searchable indices along those divisions?

Search zone: Selectively Indexing the right content

Search zone are subset of website that have been indexed separately from the rest of the site contents. When you search a search zone, you have through interaction with the site already identified yourself as a member of a particular audience or as someone searching for a particular type of information. The search zones in a site match those specific needs and results are improved retrieval performance. The user is simply less likely to retrieve irrelevant information. Also note the full site search option: sometimes it does make sense to maintain an index of the entire site, especially for user who are unsure where to look, who are doing a comprehensive leave no stones unturned search, or who just haven't had any luck searching the more narrowly defined indices.

How is search zone indexing set up? It depends on the search engine software used Most support the creation of search zone, but some provides interfaces that make this process easier, while other require you to manually provide a list of pages to index. You can create search zones in many ways.

Examples of four common approaches are:

- By content type
- By audience

- By subject
- By date

2.4.1 Grouping Content

Grouping content into the top-level categories of an information hierarchy is typically the most important and challenging process you will face. How should the content be organized? By audience or format or function? How do users currently navigate this information? How do the clients want users to navigate? Which content items should be included in which major categories? The design of information architectures should be determined by research involving members of the team and representatives from each of the major audiences. Fortunately, you don't need the least technology to conduct this research. Index cards, the 3 x 5 inch kindly you can fit in your pocket and find information any stationery store, will help you get the job done. For lack of a better name, we call this index card based approach content chunking. To try content chucking, buy a few packages of index cards and follow these steps:

1).Invite the team to generate a content wish list for the website on a set of index cards.

2).Instruct them to write down one content item per card.

3).Ask each member of the group or the group as a whole to organize cards into piles of related content items and assign labels to each pile.

4).Record the results of each and then move on to the next.

5).Repeat this exercise with representative members and groups of the organization and intended audiences.

6).Compare and contrast the results of each.

Analysis of the results should influence the information architecture of the web site.

This card based content chunking process can be performed corroboratively where people must reach consensus on the organization of information. Alternatively, individuals can sort the cards alone and record the results. The biggest problem with shuffling index cards is that it can be time consuming. Involving clients, colleagues and future users in the exercise and analyzing the sometimes confusing results takes time. Some of this content chunking can be accomplished through the wish list process as noted earlier. However, the major burden of content chunking responsibility often falls to the information architect in the conceptual design phase.

2.5 Conceptual design

Blueprints

What do you mean by blueprint? Blueprints are the architect's tool of choice for performing the transformation for chaos in to order. Blueprints show the relationship between pages and other content components and can be used to portray organization, navigation and labeling systems. They are often referred to as sitemaps and do in fact have much information common with those supplemental navigation systems. Both the diagram and the navigation system display the shape of the information space information overview, functioning as a condensed map for site developers and users, respectively

High -level Architecture blueprints

High level architecture blueprints are often created by information architects as part of a top down information architecture process. The very act shaping ideas in to the more structure of a blueprint forces you to become realistic and practical. During the design phase, high level blueprints are most useful for exploring primary organization schemes and approaches. High level blueprints map out the organization and labeling of major areas. Usually beginning with a bird's eye view from the main page of the website.

Creating High -Level Architecture Blue prints

These blueprints can be created by hand, but diagramming software such as Visio or OmniGraffle are preferred. These tools not only help to quickly layout the architecture Blue prints, but can also help with site implementation and administration.

Some Important points:

1). Blueprints focus on major areas and structure of site ignoring many navigation details and page level details.

2). Blueprints are excellent tools for explaining your architectural approaches.

3). Presenting blueprints information person allows you to immediately answer the questions and address client concerns as well as to explore new ideas while they are fresh in your mind and the client's.

4). As you create blueprint it is important to avoid getting locked into a particular type of layout.

5). If a meeting isn't possible, you can accompany blueprints with descriptive test based documents that anticipate and answer the most likely documents.

Keeping Blueprints Simple

As a project moves from strategy to design to implementation, blueprints become more utilitarian. They need be produced and modified quickly and often draw input front increasing number of perspectives, ranging from visual designers to editors to programmers. Those team members need to be able to understand the architecture. So it's important to develop a simple condensed vocabulary of objects that can explain in a brief legend.

2.5.1 Architectural Page Mockups

Information architecture blueprints are most useful for presenting a bird's eye view of the web site. However they to not work well for helping people to envision the contents of any particular page. They are also not straightforward enough for most graphic designers to work from. In Fact no single format perfect job of conveying all aspects of information architecture to all audiences. Because information architectures are multi dimensional, it's important to show them information multiple ways. For these reasons Architectural page mockup are useful tools during conceptual design for complimenting the blueprint view of the site mockups are quick and dirty textual documents that show the content and links of major pages on the website. They enable you to clearly (yet inexpensively) communicate the implications of the architecture at the page level. They are also extremely useful when used in conjunction with scenarios. They help people to see the site in action before any code is written. Finally, they can be employed in some basic usability tests to see if users actually follow the scenarios as you expect. Keep in mind that you only need to mockup major pages of the web site. These mockups and the designs that derive from them can serve as templates for design of subsidiary pages. The mockups are easier to read than blueprints. By integrating aspects of the organizational labeling, and navigation systems in to one view they will help your colleagues to understand the architecture. In laying out the content on a page mockup, you should try to show the logical visual grouping of content items. Placing a content group at the top of the page or using a larger font size indicates the relative importance of that content.

While the graphic designer will make the final and more detailed layout decisions you can make a good start with these mockups.

2.5.2 Design Sketches

Once you've evolved high-level blueprints and architectural page mockups, you're ready to collaborate with your graphic designer to create deign sketches on paper of major pages in the web site. In the research phase the design team has begun to develop a sense of the desired graphic identity or look and feel. The technical team has assessed the information technology infrastructure of the organization and the platform limitations of the intended audiences. They understand what's possible with respect to features such as dynamic content management and interactivity. And of course the architect has designed the high-level information structure for the site. Design sketches are a great way to pool the collective knowledge of these three teams in a first attempt at interface design for the top level pages of the site. This in a wonderful opportunity for interdisciplinary user interface design using the architectural mocks ups as a guide; the designer begins sketching pates of the site on sheets of paper. As the designer sketches each page questions arise that must be discussed.

Here is a sample sketching session dialog:

Programmer: I like what you're doing with the layout of the main page, but I'd like to do something more interesting with the navigation system.

Designer: Can we implement the navigation system using pull down menus? Does that make sense architecturally?

Architect: That might work but it would be difficult to show context in the hierarchy. How about a tear-|way table of contents feature? We've had pretty good reactions to that type of approach front users in the past.

Programmer: We can certainly go with that approach from a purely technical perspective. How would a tear away table of contents look? Can you sketch it for us? I'd like to do a quick and dirty prototype. These sketches allow rapid iteration and intense collaboration.

2.6 Website Development Phases

There are numerous steps in the web site design and development process. From gathering initial information, to the creation of your web site, and finally to maintenance to keep your web site up to date and current.

The exact process will vary slightly from designer to designer, but the basics are generally the same as:

- Information Gathering
- Planning
- Design
- Development
- Testing and Delivery
- Maintenance

2.6.1 Information Gathering

The first step in designing a successful web site is to gather information. Many things need to be taken into consideration when the look and feel of your site is created.

This first step is actually the most important one, as it involves a solid understanding of the company it is created for. It involves a good understanding of you - what your business goals and dreams are, and how the web can be utilized to help you achieve those goals.

It is important that your web designer start off by asking a lot of questions to help them understand your business and your needs in a web site.

Certain things to consider are:

Purpose

What is the purpose of the site? Do you want to provide information, promote a service, sell a product…?

Goals

What do you hope to accomplish by building this web site? Two of the more common goals are either to make money or share information.

Target Audience

Is there a specific group of people that will help you reach your goals? It is helpful to picture the "ideal" person you want to visit your web site. Consider their age, sex or interests - this will later help determine the best design style for your site.

Content

What kind of information will the target audience be looking for on your site? Are they looking for specific information, a particular product or service, online ordering…?

2.6.2 Planning

Using the information gathered from phase one, it is time to put together a plan for your web site. This is the point where a site map is developed.

The site map is a list of all main topic areas of the site, as well as sub-topics, if applicable. This serves as a guide as to what content will be on the site, and is essential to developing a consistent, easy to understand navigational system. The end-user of the web site - aka your customer - must be kept in mind when designing your site. These are, after all, the people who will be learning about your service or buying your product. A good user interface creates an easy to navigate web site, and is the basis for this.

During the planning phase, your web designer will also help you decide what technologies should be implemented. Elements such as interactive forms, ecommerce, flash, etc. are discussed when planning your web site.

2.6.3 Designing

Drawing from the information gathered up to this point, it's time to determine the look and feel of your site.

Target audience is one of the key factors taken into consideration. A site aimed at teenagers, for example, will look much different than one meant for a financial institution. As part of

the design phase, it is also important to incorporate elements such as the company logo or colors to help strengthen the identity of your company on the web site.

Your web designer will create one or more prototype designs for your web site. This is typically a .jpg image of what the final design will look like. Often times you will be sent an email with the mock-ups for your web site, while other designers take it a step further by giving you access to a secure area of their web site meant for customers to view work in progress.

Either way, your designer should allow you to view your project throughout the design and development stages. The most important reason for this is that it gives you the opportunity to express your likes and dislikes on the site design.

In this phase, communication between both you and your designer is crucial to ensure that the final web site will match your needs and taste. It is important that you work closely with your designer, exchanging ideas, until you arrive at the final design for your web site.

Then development can begin…

2.6.4 Development

The developmental stage is the point where the web site itself is created. At this time, your web designer will take all of the individual graphic elements from the prototype and use them to create the actual, functional site.

This is typically done by first developing the home page, followed by a "shell" for the interior pages. The shell serves as a template for the content pages of your site, as it contains the main navigational structure for the web site. Once the shell has been created, your designer will take your content and distribute it throughout the site, in the appropriate areas.

Elements such as interactive contact forms, flash animations or ecommerce shopping carts are implemented and made functional during this phase, as well.

This entire time, your designer should continue to make your in-progress web site available to you for viewing, so that you can suggest any additional changes or corrections you would like to have done.

On the technical front, a successful web site requires an understanding of front-end web development. This involves writing valid XHTML / CSS code that complies to current web standards, maximizing functionality, as well as accessibility for as large an audience as possible.

This is tested in the next phase…

2.6.5 Testing

At this point, your web designer will attend to the final details and test your web site. They will test things such as the complete functionality of forms or other scripts, as well last testing for last minute compatibility issues (viewing differences between different web browsers), ensuring that your web site is optimized to be viewed properly in the most recent browser versions.

A good web designer is one who is well versed in current standards for web site design and development. The basic technologies currently used are XHTML and CSS (Cascading Style Sheets). As part of testing, your designer should check to be sure that all of the code written for your web site validates. Valid code means that your site meets the current web development standards - this is helpful when checking for issues such as cross-browser compatibility as mentioned above.

Once you give your web designer final approval, it is time to deliver the site. An FTP (File Transfer Protocol) program is used to upload the web site files to your server. Most web designers offer domain name registration and web hosting services as well. Once these accounts have been setup, and your web site uploaded to the server, the site should be put through one last run-through. This is just precautionary, to confirm that all files have been uploaded correctly, and that the site continues to be fully functional.

This marks the official launch of your site, as it is now viewable to the public.

2.6.6 Maintenance

The development of your web site is not necessarily over, though. One way to bring repeat visitors to your site is to offer new content or products on a regular basis. Most web designers will be more than happy to continue working together with you, to update the

information on your web site. Many designers offer maintenance packages at reduced rates, based on how often you anticipate making changes or additions to your web site.

If you prefer to be more hands on, and update your own content, there is something called a CMS (Content Management System) that can be implemented to your web site. This is something that would be decided upon during the Planning stage. With a CMS, your designer will utilize online software to develop a database driven site for you.

A web site driven by a CMS gives you the ability to edit the content areas of the web site yourself. You are given access to a back-end administrative area, where you can use an online text editor (similar to a mini version of Microsoft Word). You'll be able to edit existing content this way, or if you are feeling more adventurous, you can even add new pages and content yourself. The possibilities are endless!

It's really up to you as far as how comfortable you feel as far as updating your own web site. Some people prefer to have all the control so that they can make updates to their own web site the minute they decide to do so. Others prefer to hand off the web site entirely, as they have enough tasks on-hand that are more important for them to handle directly.

That's where the help of your web designer comes in, once again, as they can take over the web site maintenance for you - one less thing for you to do is always a good thing in these busy times!

Other maintenance type items include SEO (Search Engine Optimization) and SES (Search Engine Submission). This is the optimization of you web site with elements such as title, description and keyword tags which help your web site achieve higher rankings in the search engines. The previously mentioned code validation is something that plays a vital role in SEO, as well.

There are a lot of details involved in optimizing and submitting your web site to the search engines - enough to warrant its own post. This is a very important step, because even though you now have a web site, you need to make sure that people can find it!

2.7 Check Your Progress

1. Some users information needs are clearly defined and have a single correct answer, this type of searching is called ……...
2. Some users know how to phrase their question, but don't know exactly what they are hoping to find and are really just exploring and trying to learn more, this type of searching is called ……...
3. Fielded Searching option comes under …….Search Interface.
4. …….and browsing systems should be closely integrated.
5. Search zone are subset of ……... that have been indexed separately from the rest of the site contents.
6. ……are the architect's tool of choice for performing the transformation for chaos in to order.
7. …….. are excellent tools for explaining your architectural approaches.
8. Purpose of website is defined in ……phase of development of website.
9. Development phase comes after …….in development of website.
10. If you prefer to be more hands on, and update your own content, there is something called a …….

2.8 Summary

Assuming you have decided to implement a Searching system for your website. It's important to understand how users really search before designing it.

Assuming you have decided to implement a Searching system for your website. It's important to understand how users really search before designing it.

- Known item searching
- Existence searching
- Exploratory searching
- Comprehensive Searching (Research)

There are two models of searching systems:
1). In the first and older model user express their information need as query that they enter in a search interface. They may do so using a specialized search language.
2). In the second model users express the information need information the natural language like English.
Support Different Modes of Searching

- Simple / Basic search interface
- Advanced search Interface

In fact users often don't know if they need to search or browse in the first place. Therefore, these respective systems shouldn't live in isolation from one another. The search/browse approach can be extended by making search and browse options available on the search result page as well, especially on null results pages when a user might be at a dead end and needs to be gently led back to the process of iterative searching and browsing before frustration sets in.
Searching only works well when the stuff that's being searched is the same as the stuff that users want. This means you may not want to index the entire site.

Search zone are subset of website that have been indexed separately from the rest of the site contents. When you search a search zone, you have through interaction with the site already identified yourself as a member of a particular audience or as someone searching for a particular type of information.

Blueprints are the architect's tool of choice for performing the transformation for chaos in to order. Blueprints show the relationship between pages and other content components and can be used to portray organization, navigation and labeling systems. High level architecture blueprints are often created by information architects as part of a top down information architecture process. The very act shaping ideas in to the more structure of a blueprint forces you to become realistic and practical.
There are numerous steps in the web site design and development process. From gathering initial information, to the creation of your web site, and finally to maintenance to keep your web site up to date and current.

The exact process will vary slightly from designer to designer, but the basics are generally the same as:

- Information Gathering
- Planning
- Design
- Development
- Testing and Delivery
- Maintenance

2.9 Self-Assessment Test

Q.1 Explain Searching System and Its importance.

Q.2. Briefly explain the concept designing Searching System.

Q.3 What do you understand by Architectural Page Mockup? Explain.

Q.4 Differentiate between Conceptual Design and Architectural Page Mockup.

Q.5 Explain different phases of website development.

2.10 Answers to check your progress

1. Known item searching
2. Exploratory searching
3. Advanced
4. Searching
5. Website
6. Blueprints
7. Blueprints
8. Information Gathering
9. Designing Phase
10. CMS (Content Management System)

2.11 References / Suggested Readings

1. Thomas A Powell, HTML-The Complete Reference,Tata McGraw Hill.
2. Scott Guelich, Shishir Gundavaram, Gunther Birzniek; CGI Programming with Perl 2/e. O'Reilly.
3. Doug Tidwell, James Snell, Pavel Kulchenko; Programming Web Services with SOAP, O'Reilly.
4. Pardi, XML in Action, Web Technology, PHI.
5. Yong, XML Step by Step, PHI.
6. Aaron Weiss, Rebecca Taply, Kim Daniels, Stuven Mulder, Jeff Kaneshki, Web Authoring Desk Reference, Techmedia Publications.
7. HTML The complete Reference, TMH

SUBJECT: WEB DESIGNING	
COURSE CODE: MCA-12	**AUTHOR: DR. SUDHANSHU GAUR**
LESSON NO. 3	

Web Design and HTML

STRUCTURE

3.0 LEARNING OBJECTIVE

After going through this unit, you will be able to:

- Learn about good Web Design.
- Learn about Basic Structure of HTML.
- Different core attributes of HTML
- Learn about Different Layout in HTML.
- Learn about How to link documents in HTML.

3.1 INTRODUCTION

This chapter give a brief idea about DHTML. We will have a look at web design process. We will learn how to make web design better and better.

We will create website using HTML pages. We will learn about the basic structure of HTML. Different core attributes of HTML also covered in this chapter.

We will create tables in HTML and we also learn how to link documents in HTML.

3.2 DHTML

DHTML stands for Dynamic HTML. It is NOT a language or a web standard. DHTML is the art of combining HTML, JavaScript, DOM, and CSS.

According to the World Wide Web Consortium (W3C):

"Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."

DHTML Technologies

Below is a listing of DHTML technologies:

HTML 4.0

The HTML 4.0 standard has rich support for dynamic content like:

- HTML supports JavaScript
- HTML supports the Document Object Model (DOM)
- HTML supports HTML Events
- HTML supports Cascading Style Sheets (CSS)

DHTML is about using these features to create dynamic and interactive web pages.

JavaScript

JavaScript is the scripting standard for HTML.

DHTML is about using JavaScript to control, access and manipulate HTML elements.

HTML DOM

The HTML DOM is the W3C standard Document Object Model for HTML. It defines a standard set of objects for HTML, and a standard way to access and manipulate them.

DHTML is about using the DOM to access and manipulate HTML elements.

CSS

CSS is the W3C standard style and layout model for HTML. It allows web developers to control the style and layout of web pages. HTML 4 allows dynamic changes to CSS.

DHTML is about using JavaScript and DOM to change the style and positioning of HTML elements.

3.3 Web Designing

Web design is the creation of Web pages and sites using HTML, CSS, JavaScript and other Web languages. Web design is just like design in general: it is the combination of lines,

shapes, texture, and color to create an aesthetically pleasing or striking look. Web design is the work of creating design for Web pages.

The process of designing Web pages, Web sites, Web applications or multimedia for the Web may utilize multiple disciplines, such as animation, authoring, communication design, corporate identity, graphic design, human-computer interaction, information architecture, interaction design, marketing, photography, search engine optimization and typography.

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

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

3.3.1 Good Web Design

Web design can be deceptively difficult, as it involves achieving a design that is both usable and pleasing, delivers information and builds brand, is technically sound and visually coherent.

Principles for good Web design:

Precedence (Guiding the Eye)

Good Web design, perhaps even more than other type of design, is about information. One of the biggest tools in your arsenal to do this is precedence. When navigating a good design, the user should be led around the screen by the designer. I call this precedence, and it's about how much visual weight different parts of your design have.

A simple example of precedence is that in most sites, the first thing you see is the logo. This is often because it’s large and set at what has been shown in studies to be the first place people look (the top left). his is a good thing since you probably want a user to immediately know what site they are viewing.

But precedence should go much further. You should direct the user's eyes through a sequence of steps. For example, you might want your user to go from logo/brand to a primary positioning statement, next to a punchy image (to give the site personality), then to the main body text, with navigation and a sidebar taking a secondary position in the sequence.

What your user should be looking at is up to you, the Web designer, to figure out.

To achieve precedence you have many tools at your disposal:

Position — Where something is on a page clearly influences in what order the user sees it.

Color — Using bold and subtle colors is a simple way to tell your user where to look.

Contrast — Being different makes things stand out, while being the same makes them secondary.

Size — Big takes precedence over little (unless everything is big, in which case little might stand out thanks to Contrast)

Design Elements — if there is a gigantic arrow pointing at something, guess where the user will look?

Spacing

When I first started designing, I wanted to fill every available space up with stuff. Empty space seemed wasteful. In fact, the opposite is true.

Spacing makes things clearer. In Web design there are three aspects of space that you should be considering:

Line Spacing

When you lay text out, the space between the lines directly affects how readable it appears. Too little space makes it easy for your eye to spill over from one line to the next, too much space means that when you finish one line of text and go to the next your eye can get lost. So you need to find a happy medium. You can control line spacing in CSS with the 'line-

height' selector. Generally, I find the default value is usually too little spacing. Line Spacing is technically called leading (pronounced ledding), which derives from the process that printers used to use to separate lines of text in ye old days — by placing bars of lead between the lines.

Padding

Generally speaking text should never touch other elements. Images, for example, should not be touching text, neither should borders or tables. Padding is the space between elements and text. The simple rule here is that you should always have space there. There are exceptions of course, in particular if the text is some sort of heading/graphic or your name is David Carson :-) But as a general rule, putting space between text and the rest of the world makes it infinitely more readable and pleasant.

White Space

First of all, white space doesn't need to be white. The term simply refers to empty space on a page (or negative space as it's sometimes called). White space is used to give balance, proportion and contrast to a page. A lot of white space tends to make things seem more elegant and upmarket, so for example if you go to an expensive architect site, you'll almost always see a lot of space. If you want to learn to use whitespace effectively, go through a magazine and look at how adverts are laid out. Ads for big brands of watches and cars and the like tend to have a lot of empty space used as an element of design.

Navigation

One of the most frustrating experiences you can have on a Web site is being unable to figure out where to go or where you are. I'd like to think that for most Web designers, navigation is a concept we've managed to master, but I still find some pretty bad examples out there. There are two aspects of navigation to keep in mind:

Navigation — Where can you go?

There are a few commonsense rules to remember here. Buttons to travel around a site should be easy to find - towards the top of the page and easy to identify. They should look like navigation buttons and be well described. The text of a button should be pretty clear as to where it's taking you. Aside from the common sense, it's also important to make

navigation usable. For example, if you have a rollover sub-menu, ensuring a person can get to the sub-menu items without losing the rollover is important. Similarly changing the color or image on rollover is excellent feedback for a user.

Orientation — Where are you now?

There are lots of ways you can orient a user so there is no excuse not to. In small sites, it might be just a matter of a big heading or a 'down' version of the appropriate button in your menu. In a larger site, you might make use of bread crumb trails, sub-headings and a site map for the truly lost.

Design to Build

Life has gotten a lot easier since Web designers transitioned to CSS layouts, but even now it's still important to think about how you are going to build a site when you're still in Photoshop. Consider things like:

Can it actually be done?

You might have picked an amazing font for your body copy, but is it actually a standard HTML font? You might have a design that looks beautiful but is 1100px wide and will result in a horizontal scroller for the majority of users. It's important to know what can and can't be done, which is why I believe all Web designers should also build sites, at least sometimes.

What happens when a screen is resizes?

Do you need repeating backgrounds? How will they work? Is the design centered or left-aligned?

Are you doing anything that is technically difficult?

Even with CSS positioning, some things like vertical alignment are still a bit painful and sometimes best avoided.

Could small changes in your design greatly simplify how you build it?

Sometimes moving an object around in a design can make a big difference in how you have to code your CSS later. In particular, when elements of a design cross over each other, it adds a little complexity to the build. So if your design has, say three elements and each element is completely separate from each other, it would be really easy to build. On the other hand, if all three overlap each other, it might still be easy, but will probably be a bit more complicated. You should find a balance between what looks good and small changes that can simplify your build.

For large sites, particularly, can you simplify things?

There was a time when I used to make image buttons for my sites. So if there was a download button, for example, I would make a little download image. In the last year or so, I've switched to using CSS to make my buttons and have never looked back. Sure, it means my buttons don't always have the flexibility I might wish for, but the savings in build time from not having to make dozens of little button images are huge.

Typography

Text is the most common element of design, so it's not surprising that a lot of thought has gone into it. It's important to consider things like:

Font Choices — Different types of fonts say different things about a design. Some look modern, some look retro. Make sure you are using the right tool for the job.

Font sizes —Years ago it was trendy to have really small text. Happily, these days people have started to realize that text is meant to be read, not just looked at. Make sure your text sizes are consistent, large enough to be read, and proportioned so that headings and sub-headings stand out appropriately.

Spacing — As discussed above, spacing between lines and away from other objects is important to consider. You should also be thinking about spacing between letters, though on the Web this is of less importance, as you don't have that much control.

Line Length — There is no hard and fast rule, but generally your lines of text shouldn't be too long. The longer they are, the harder they are to read. Small columns of text work much better (think about how a newspaper lays out text).

Color — One of my worst habits is making low-contrast text. It looks good but doesn't read so well, unfortunately. Still, I seem to do it with every Web site design I've ever made, tsk tsk tsk.

Paragraphing — Before I started designing, I loved to justify the text in everything. It made for nice edges on either side of my paragraphs. Unfortunately, justified text tends to create weird gaps between words where they have been auto-spaced. This isn't nice for your eye when reading, so stick to left-aligned unless you happen to have a magic body of text that happens to space out perfectly.

Usability

Web design isn’t just about pretty pictures. With so much information and interaction to be affected on a Web site, it's important that you, the designer, provide for it all. That means making your Web site design usable.

We've already discussed some aspects of usability - navigation, precedence, and text. Here are three more important ones:

Adhering to Standards

There are certain things people expect, and not giving them causes confusion. For example, if text has an underline, you expect it to be a link. Doing otherwise is not good usability practice. Sure, you can break some conventions, but most of your Web site should do exactly what people expect it to do!

Think about what users will actually do

Prototyping is a common tool used in design to actually 'try' out a design. This is done because often when you actually use a design, you notice little things that make a big difference. ALA had an article a while back called Never Use a Warning When You Mean Undo, which is an excellent example of a small interface design decision that can make life suck for a user.

Think about user tasks

When a user comes to your site what are they actually trying to do? List out the different types of tasks people might do on a site, how they will achieve them, and how easy you

want to make it for them. This might mean having really common tasks on your homepage (e.g. 'start shopping', 'learn about what we do,' etc.) or it might mean ensuring something like having a search box always easily accessible. At the end of the day, your Web design is a tool for people to use, and people don't like using annoying tools

Alignment

Keeping things lined up is as important in Web design as it is in print design. That's not to say that everything should be in a straight line, but rather that you should go through and try to keep things consistently placed on a page. Aligning makes your design more ordered and digestible, as well as making it seem more polished.

You may also wish to base your designs on a specific grid. I must admit I don't do this consciously - though obviously a site like PSDTUTS does in fact have a very firm grid structure. This year I've seen a few really good articles on grid design including Smashing Magazine's Designing with Grid-Based Approach & A List Apart's Thinking Outside The Grid. In fact, if you're interested in grid design, you should definitely pay a visit to the aptly named DesignByGrid.com home to all things griddy.

Clarity (Sharpness)

Keeping your design crisp and sharp is super important in Web design. And when it comes to clarity, it's all about the pixels.

In your CSS, everything will be pixel perfect so there's nothing to worry about, but in Photoshop it is not so. To achieve a sharp design, you have to:

* Keep shape edges snapped to pixels. This might involve manually cleaning up shapes, lines, and boxes if you're creating them in Photoshop.

Make sure any text is created using the appropriate anti-aliasing setting. I use 'Sharp' a lot.

Ensuring that contrast is high so that borders are clearly defined.

Over-emphasizing borders just slightly to exaggerate the contrast.

Consistency

Consistency means making everything match. Heading sizes, font choices, coloring, button styles, spacing, design elements, illustration styles, photo choices, etc. Everything should be themed to make your design coherent between pages and on the same page.

Keeping your design consistent is about being professional. Inconsistencies in a design are like spelling mistakes in an essay. They just lower the perception of quality. Whatever your design looks like, keeping it consistent will always bring it up a notch. Even if it's a bad design, at least make it a consistent, bad design.

The simplest way to maintain consistency is to make early decisions and stick to them. With a really large site, however, things can change in the design process. When I designed FlashDen, for example, the process took months, and by the end some of my ideas for buttons and images had changed, so I had to go back and revise earlier pages to match later ones exactly.

Having a good set of CSS stylesheets can also go a long way to making a consistent design. Try to define core tags like <h1> and <p> in such a way as to make your defaults match properly and avoid having to remember specific class names all the time.

3.4 Web Publishing Process

Once we create our website, we need to do the following to publish our website:

1). Determine a Domain Name.

2). Register the domain name.

3). Locate and register with a web host and also verify whether or not your domain name is already in use.

4). Now uploading your web pages to the host's site. For this you use FTP to transfer the site from computer to host.

5). Finally, test your site by travelling pages and hyperlinks, making sure that the site performs to your satisfaction.

3.5 Structure of HTML Document

At the foundation of every HTML file is a set of structure tags that divide an HTML file into a head section and a body section. These two sections are enclosed between an opening <HTML> tag and end with the closing <HTML> tag. Following is a simple example of an HTML file that highlights the structure tags that are required within every XHTML file.

--------- Doctype Declaration ----------

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

--------- Opening html tag ----------

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

--------- Head Section ----------

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>
```

--------- Body Section ----------

```
<body>
    My First Page.
</body>
```

--------- Closing html tag ----------

```
</html>
```

The head and body sections are contained within the opening html tag and the closing html tag.

The Doctype declaration is found at the beginning of every XHTML file. It is the only tag that can contain uppercase text.

Head Section

Contains the title of the webpage and information pertaining to the entire Web page.

Contains any “meta” tags. The meta tag shown in the example below is required in all XHTML files.

Nothing in the head section is visible in the browser window except for the title, “My Webpage” that is found between the title tags (<title>My Webpage</title>).

Body Section

Contains the file contents that are visible in the browser window. In other words, whatever you find within the body tags is what you will see in the browser window.

3.6 HTML Elements

HTML documents are defined by HTML elements. An HTML element is everything from the start tag to the end tag. The start tag is often called the opening tag. The end tag is often called the closing tag.

Start tag Element Content End tag

HTML Element Syntax

- An HTML element starts with a start tag / opening tag
- An HTML element ends with an end tag / closing tag
- The element content is everything between the start and the end tag
- Some HTML elements have empty content
- Empty elements are closed in the start tag
- Most HTML elements can have attributes

Example

<html>
<body>
This is my first paragraph
</body>
</html>

Example

The element

This is my first paragraph

The element defines a paragraph in the HTML document.

The element has a start tag and an end tag

The element content is: This is my first paragraph.

The <body> element

<body>

This is my first paragraph

<body>

The <body> element defines the body of the HTML document

The element has a start tag <body> and an end tag <body>

The element content is another HTML element (a paragraph)

The <html> element

<html>

<body>

This is my first paragraph

</body>

</html>

The <html> element defines the whole HTML document.

The element has a start tag <html> and an end tag <html>

The element content is another HTML element (the body)

Don't Forget the End Tag

Most browsers will display HTML correctly even if you forget the end tag. But forgetting the end tag can produce unexpected results or errors. Future version of HTML will not allow you to skip end tags.

Empty HTML Elements

HTML elements without content are called empty elements. Empty elements can be closed in the start tag.

 is an empty element without a closing tag (it defines a line break). Adding a slash to the start tag, like <br />, is the proper way of closing empty elements, accepted by HTML, XHTML and XML. Even if <br> works in all browsers, writing <br /> instead is more future proof.

Use Lowercase Tags

HTML tags are not case sensitive: <P> means the same as <p>. Plenty of web sites use uppercase HTML tags in their pages. World Wide Web Consortium (W3C) recommends lowercase in HTML 4, and demands lowercase tags in future versions of (X)HTML.

3.6.1 Core Attributes

HTML 4.0 has a set of four core attributes: ID, CLASS, STYLE and TITLE attribute.

ID Attribute

The ID attribute uniquely identifies an element within a document. No two elements can have the same ID value in a single document. The attribute's value must begin with a letter in the range A-Z or a-z, digits (0-9), hyphens ("-"), underscores ("_"), colons (":"), and periods (".). The value is case-sensitive.

The following example uses the ID attribute to identify each of the first two paragraphs of a document:

```
<P ID=firstparagraph>This is my first paragraph.</P>
<P ID=secondparagraph>This is my second paragaph.</P>
```

In the example, both paragraphs could have style rules associated with them through their ID attributes. The following Cascading Style Sheet defines unique colors for the two paragraphs:

```
<style type="text/css">
#firstparagraph
{
color : red;
}
#secondparagraph
{
color : blue;
```

}
</style>
Output:
This is my first paragraph.
This is my second paragraph.
A style sheet rule could be put in the head i.e.(<head>..<style>...</style>...</head>)of a document.

Class Attribute

The class attribute is used to indicate the class or classes that a tag might belong to. Like id, class is used to associate a tag with a name, so

This is the first paragraph of text.

not only names the paragraph uniquely as FirstParagraph, but also indicates that this paragraph belongs to a class grouping called important. The main use of the class attribute is to relate a group of elements to various style sheet rules. For example, a style sheet rule such as

<style type="text/css">
.important {background-color: yellow;}
</style>

would give all elements with the class attribute set to important a yellow background. Given that many elements can have the same class values, this may affect a large portion of the document.

Style Attribute

The style attribute is used to add style sheet information directly to a tag. For example,

This is the first paragraph of text.

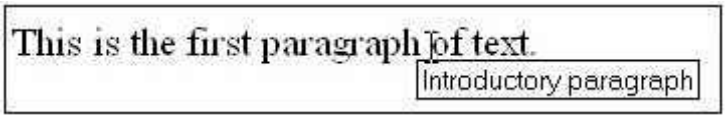
sets the font size of the paragraph to be 18 point, red text. Although the styleattribute allows CSS rules to be added to an element with ease, it is preferable to use id or class to relate a document-wide or linked style sheet.

Title Attribute

The title is used to provide advisory text about an element or its contents. In the case of

This is the first paragraph of text.

the title attribute is set to indicate that this particular paragraph is the introductory paragraph. Browsers can display this advisory text in the form of a Tooltip, as shown here:



Tooltips set with title values are often found on links, form fields, images, and anywhere where an extra bit of information is required.

The core attributes might not make a great deal of sense at this time because generally they are most useful with scripting and style sheets, but keep in mind that these four attributes are assumed with *every* tag that is introduced for the rest of this chapter.

3.6.2 Relative and Absolute Link

"A LINK IS AN ABSOLUTE LINK IF THE URL AND FILE NAME CAN BE FOUND FROM ANYWHERE ON THE WEB, NOT JUST FROM A SINGLE WEB SITE."

An absolute link specifies a fully-qualified URL; the protocol must be present in addition to a domain name, and often a file name must be included as well.

For instance:

<A HREF="http://www.compugoddess.com/index.htm">Go to CompuGoddess

The URL in this link can be browsed to *regardless of where one starts*.

A relative link specifies the name of the file to be linked to *only as it is related to the current document*.

For example, if all the files in your Web site are contained within the same directory (or folder), and you want to establish a link from page1.html to page2.html, the code on page1.html will be:

Go to page 2

This link will be valid *only* from within the same directory that page2.html is saved in.

3.6.3 Types of Lists

HTML offers authors several mechanisms for specifying lists of information. All lists must contain one or more list elements. Lists may contain:

- Unordered information.
- Ordered information.
- Definitions.

The previous list, for example, is an **unordered list**, created with the UL element:

<UL>
<LI>Unordered information.
<LI>Ordered information.
<LI>Definitions.
</UL>

An ordered list, created using the OL element, should contain information where order should be emphasized, as in a recipe:

1. Mix dry ingredients thoroughly.
2. Pour in wet ingredients.
3. Mix for 10 minutes.
4. Bake for one hour at 300 degrees.

Definition lists, created using the DL element, generally consist of a series of term/definition pairs (although definition lists may have other applications). Thus, when advertising a product, one might use a definition list:

Lower cost
The new version of this product costs significantly less than the previous one!

Easier to use
We've changed the product so that it's much easier to use!

Safe for kids
You can leave your kids alone in a room with this product and they won't get hurt (not a guarantee).

3.7 Linking in HTML

A link is the "address" to a document (or a resource) on the web.

Hyperlinks
In web terms, a hyperlink is a reference (an address) to a resource on the web. Hyperlinks can point to any resource on the web: an HTML page, an image, a sound file, a movie, etc.

Anchors
An anchor is a term used to define a hyperlink destination inside a document. The HTML anchor element <a>, is used to define both hyperlinks and anchors.
We will use the term HTML link when the <a> element points to a resource, and the term HTML anchor when the <a> elements defines an address inside a document.

HTML Link Syntax
Link text<a>
The start tag contains attributes about the link. The element content (Link text) defines the part to be displayed.

The href Attribute
The href attribute defines the link "address".
This <a> element defines a link to onlinemca.com:
Visit onlinemca<a>

The target Attribute
The target attribute defines where the linked document will be opened. The code below will open the document in a new browser window:
Visit onlinemca

3.8 Images and Anchors

If you want to make an image work as a link, the method is exactly the same as with texts. You simply place the <a href> and the tags on each side of the image.

Below is the HTML code used to make the image work as a link to a page
<img src="xxx.gif">

If you haven't entered a border setting you will see a small border around the image after turning it into a link. To turn off this border, simply add border="0" to the <img> tag:
<img src="xxx.gif" border="0">

Images that work as links can show a popup text when you place the mouse over it. This is done with the alt property in the <img> tag.
<img src="xxx.gif" border="0" alt="Link to this page">

3.8.1 Anchor Attribute

The <a> tag defines an anchor. An anchor can be used in two ways:
1. To create a link to another document, by using the href attribute.
2. To create a bookmark inside a document, by using the name attribute.

The a element is usually referred to as a link or a hyperlink. The most important attribute of the a element is the href attribute, which indicates the link's destination.
By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

3.8.2 Image Maps

In HTML and XHTML, an image map is a list of coordinates relating to a specific image, created in order to hyperlink areas of the image to various destinations (as opposed to a normal image link, in which the entire area of the image links to a single destination). For example, a map of the world may have each country hyperlinked to further information about that country. The intention of an image map is to provide an easy way of linking various parts of an image without dividing the image into separate image files.

It is possible to create image maps by hand, using a text editor, however doing so requires that the web designer knows how to code HTML and also requires them to know the coordinates of the areas that they wish to place over the image. As a result, most image maps coded by hand are simple polygons.

Because creating image maps in a text editor requires much time and effort, there are many applications that allow the web designer to quickly and easily create image maps much as they would create shapes in a vector graphics editor. Examples of these are Adobe's Dreamweaver or KImageMapEditor (for KDE), and the imagemap plugin found in GIMP
.

The <map> tag

The <map> tag is used to define a client-side image-map. An image-map is an image with clickable areas.

The name attribute is required in the map element. This attribute is associated with the <img>'s usemap attribute and creates a relationship between the image and the map.

The map element contains a number of area elements, that defines the clickable areas in the image map.

3.8.3 Semantic Linking Meta Information

The Semantic Web is an evolving development of the World Wide Web in which the meaning (semantics) of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content. It derives from World Wide Web Consortium director Sir Tim Berners-Lee's vision of the Web as a universal medium for data, information, and knowledge exchange.

At its core, the semantic web comprises a set of design principles, collaborative working groups, and a variety of enabling technologies. Some elements of the semantic web are expressed as prospective future possibilities that are yet to be implemented or realized. Other elements of the semantic web are expressed in formal specifications. Some of these include Resource Description Framework (RDF), a variety of data interchange formats (e.g. RDF/XML, N3, Turtle, N-Triples), and notations such as RDF Schema (RDFS) and the Web Ontology Language (OWL), all of which are intended to provide a formal description of concepts, terms, and relationships within a given knowledge domain.

3.8.4 Image Preliminaries

In HTML, images are defined with the <img> tag.

The <img> tag is empty, which means that it contains attributes only and it has no closing tag. To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display on your page.

Syntax

The URL points to the location where the image is stored.

3.8.5 Images as Buttons

Image buttons have the same effect as submit buttons. When a visitor clicks an image button the form is sent to the address specified in the action setting of the <form> tag.

3.9 Introduction to Layout

Layout is the compilation of text and graphics on a page. Everywhere on the Web you will find pages that are formatted like newspaper pages using HTML columns. You may have noticed that many websites have multiple columns in their layout - they are formatted like a magazine or newspaper. Many websites achieved this HTML layout using tables.

Text Layout

These tags will let you control the layout.

HTML	EXPLANATION
text	Adds a paragraph break after the text.
text	Left justify text in paragraph.
text	Center text in paragraph.
text	Right justify text in paragraph.
 	Adds a single line break where the tag is.

3.9.1 Layout with Tables

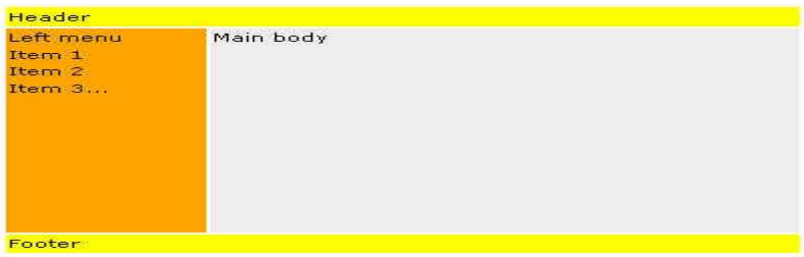
Tables have been a popular method for achieving advanced layouts in HTML. Generally, this involves putting the whole web page inside a big table. This table has a different column or row for each main section.

For example, the following HTML layout example is achieved using a table with 3 rows and 2 columns - but the header and footer column spans both columns (using the colspan attribute):

This HTML code...

```
<table width="400px" border="0">
<tr>
<td colspan="2" style="background-color: yellow;">
Header
</td>
</tr>
<tr>
<td style="background-color:orange;width:100px;text-align:top;">
Left menu<br />
Item 1<br />
Item 2<br />
Item 3...
</td>
<td style="background-color:#eeeeee;height:200px;width:300px;text-align:top;">
Main body
</td>
</tr>
<tr>
<td colspan="2" style="background-color: yellow;">
Footer
</td>
</tr>
</table>
```

Layout



3.10 Advanced Layout

3.10.1 HTML Form and Form Control

HTML Forms

A form is simply an area that can contain form fields.

Form fields are objects that allow the visitor to enter information - for example text boxes, drop-down menus or radio buttons.

When the visitor clicks a submit button, the content of the form is usually sent to a program that runs on the server. However, there are exceptions.

Javascript is sometimes used to create magic with form fields. An example could be when turning options in a drop-down menu into normal links.

The <form> tag

When a form is submitted, all fields on the form are being sent.

The <form> tag tells the browser where the form starts and ends. You can add all kinds of HTML tags between the <form> and <form> tags. This means that a form can easily include a table or an image along with the form fields mentioned on the next page.

These fields can be added to your forms:

- Text field
- Password field
- Hidden field
- Text area
- Check box
- Radio button
- Drop-down menu
- Submit button
- Reset button
- Image button

Form Control

These fields can be added to your forms:

- Text field
- Password field
- Hidden field
- Text area

- Check box
- Radio button
- Drop-down menu
- Submit button
- Reset button
- Image button

Text Fields

Text fields are used when you want the user to type letters, numbers, etc. in a form.

```
<form>
    First name:
    <input type="text" name="firstname" />
    <br />
    Last name:
    <input type="text" name="lastname" />
    <form>
```

How it looks in a browser:

firstname 

lastname 

Radio Buttons

Radio Buttons are used when you want the user to select one of a limited number of choices.

```
<form>
    <input type="radio" name="sex" value="male" /> Male
    <br />
    <input type="radio" name="sex" value="female" /> Female
    </form>
```

How it looks in a browser:

 Male

 Female

Checkboxes

Checkboxes are used when you want the user to select one or more options of a limited number of choices.

```
<form>
    I have a bike:
    <input type="checkbox" name="vehicle" value="Bike" />
    <br />
    I have a car:
    <input type="checkbox" name="vehicle" value="Car" />
    <br />
    I have an plane:
    <input type="checkbox" name="vehicle" value="Airplane" />
    </form>
```

How it looks in a browser:

I have a bike:

I have a car:

I have a plane:

3.10.2 Frames

With frames, user can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- The web developer must keep track of more HTML documents
- It is difficult to print the entire page

The frames can be created by using various tags as mentioned below:

The Frameset Tag

- The <FRAMESET> tag defines how to divide the window into frames
- Each frameset defines a set of rows or columns
- The values of the rows/columns indicate the amount of screen area each row/column will occupy

The Frame Tag

- The <FRAME> tag defines what HTML document to put into each frame.

In the example below have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The HTML document "frame_a.htm" is put into the first column, and the HTML document "frame_b.htm" is put into the second column:

```
<FRAMESET cols="25%,75%">
  <FRAME src="frame_a.htm">
  <FRAME src="frame_b.htm">
</FRAMESET>
```

The frameset column size value can also be set in pixels (cols="200,500"), and one of the columns can be set to use the remaining space (cols="25%,*").

If a frame has visible borders, the user can resize it by dragging the border. To prevent a user from doing this, you can add noresize="noresize" to the <FRAME> tag.
Add the <NOFRAMES> tag for browsers that do not support frames.

One cannot use the <BODY></BODY> tags together with the <FRAMESET></FRAMESET> tags! However, if a <NOFRAMES> tag is added containing some text for browsers that do not support frames, user will have to enclose the text in <BODY></BODY> tags!

Attributes used in <FRAMESET> tag:

rows. Defines the number and size of rows in a frameset.
cols. Defines the number and size of columns in a frameset.

Attributes used in <FRAME> tag:

frameborder. (Value: 0 or 1). Specifies whether or not to display border around the frame.
name. Defines a unique name for the frame (to use in scripts).

noresize. (Value: noresize). When set to noresize the user cannot resize the frame.
scrolling. (Value: yes, no or auto). Determines scrollbar action.

3.10.2.1 Targeting in Frames

Frames can be set as a target for any HTML document to be displayed with reference to any hyperlink.

For instance if we are having two frames in a Web page and wish the links in a frame to be opened in second frame, then we can do it by using target attribute in the anchor tag in the HTML document.
<a href=”home.asp” target=”showframe”>

Where showframe is the name of a frame set in the <FRAME> element for the particular frame.

3.11 Check Your Progress

1. DHTML stands for …….
2. DHTML is the art of combining HTML, ……, DOM, and CSS.
3. W3C stands for…….
4. CSS stands for……...
5. Full form of XHTML is …….
6. HTML language is case ……. language.
7. HTML 4.0 has a set of four core attributes: ID, CLASS, ……and TITLE attribute.
8. A ……. link specifies the name of the file to be linked to only as it is related to the current document.
9. Definition lists, created using the …….. element in HTML
10. A …….is a reference (an address) to a resource on the web.
11. An ……is a term used to define a hyperlink destination inside a document.
12. The ……. tag defines how to divide the window into frames.

3.12 Summary

DHTML stands for Dynamic HTML. It is NOT a language or a web standard. DHTML is the art of combining HTML, JavaScript, DOM, and CSS.
Web design is the creation of Web pages and sites using HTML, CSS, JavaScript and other Web languages. Web design is just like design in general: it is the combination of lines, shapes, texture, and color to create an aesthetically pleasing or striking look. Web design is the work of creating design for Web pages.

Principles for good Web design:

- Precedence (Guiding the Eye)
- Line Spacing
- Navigation
- Design to Build
- Typography
- Usability
- Alignment
- Clarity (Sharpness)
- Consistency

HTML Basic Structure

<html>
<head>……. </head>
<body>…… </body>
</html>

HTML 4.0 has a set of four core attributes: ID, CLASS, STYLE and TITLE attribute.
An absolute link specifies a fully-qualified URL; the protocol must be present in addition to a domain name, and often a file name must be included as well.
A relative link specifies the name of the file to be linked to *only as it is related to the current document.*

Lists may contain:

- Unordered information.
- Ordered information.
- Definitions.

In web terms, a hyperlink is a reference (an address) to a resource on the web. Hyperlinks can point to any resource on the web: an HTML page, an image, a sound file, a movie, etc.

The <form> tag tells the browser where the form starts and ends. You can add all kinds of HTML tags between the <form> and <form> tags. This means that a form can easily include a table or an image along with the form fields mentioned on the next page.

These fields can be added to your forms:

- Text field
- Password field
- Hidden field
- Text area
- Check box
- Radio button
- Drop-down menu
- Submit button
- Reset button
- Image button

3.13 Self-Assessment Test

Q.1 Explain principles for good web design.

Q.2. Describe web publishing process.

Q.3. Explain Basic structure of HTML.

Q.4 Describe Anchor tag and Image Tag with example.

Q.5 Explain Table Tag along with its attributes.

Q.6 Explain Frame and Frameset Tag with example.

3.14 Answers to check your progress

1. Dynamic HTML
2. JavaScript
3. World Wide Web Consortium
4. Cascading Style Sheet
5. Extensible HTML
6. Insensitive

7. STYLE
8. Relative
9. DL
10. Hyperlink
11. Anchor
12. <FRAMESET>

3.15 References / Suggested Readings

1. Thomas A Powell, HTML-The Complete Reference,Tata McGraw Hill.
2. Scott Guelich, Shishir Gundavaram, Gunther Birzniek; CGI Programming with Perl 2/e. O'Reilly.
3. Doug Tidwell, James Snell, Pavel Kulchenko; Programming Web Services with SOAP, O'Reilly.
4. Pardi, XML in Action, Web Technology, PHI.
5. Yong, XML Step by Step, PHI.
6. Aaron Weiss, Rebecca Taply, Kim Daniels, Stuven Mulder, Jeff Kaneshki, Web Authoring Desk Reference, Techmedia Publications.
7. HTML The complete Reference, TMH

SUBJECT: WEB DESIGNING	
COURSE CODE: MCA-12	**AUTHOR: DR. SUDHANSHU GAUR**
LESSON NO. 4	

HTML, Stylesheet and XML

STRUCTURE

4.0 LEARNING OBJECTIVE

After going through this chapter, you will be able to:

- Gain the concept Various types of Media with HTML support.
- Learn about Various style sheets.
- Gain the concept of CSS.
- Know about the overview of XML
- Understand the relationship between XML, HTML and SGML.

4.1 INTRODUCTION

This chapter cover different types of media that HTML supports. HTML support Audio, Video and other Binary formats. Audio, Video can be further divided into many categories. We will define style sheet and categorized it into various categories. It can be divided into three categories Inline, Internal and External.

External stylesheet is also called Cascading Style Sheet. We will also describe the linking in style sheet. We will describe positioning in style sheet.

We will take a look at XML and we will compare XML with HTML and SGML.

4.2 HTML & Media Types

Multimedia is everything you can hear or see: texts, books, pictures, music, sounds, CDs, videos, DVDs, Records, Films, and more.
Multimedia comes in many different formats. On the Internet you will find many of these elements embedded in web pages, and today's web browsers have support for a number of multimedia formats.

Multimedia Formats

Multimedia elements (like sounds or videos) are stored in media files. The most common way to discover the media type is to look at the file extension.

When a browser sees the file extensions .htm or .html, it will assume that the file is an HTML page. The .xml extension indicates an XML file, and the .css extension indicates a style sheet. Picture formats are recognized by extensions like .gif and .jpg.

Browser Support

The first Internet browsers had support for text only, and even the text support was limited to a single font in a single color, and little or nothing else. Then came web browsers with support for colors, fonts and text styles, and the support for pictures was added.

The support for sounds, animations and videos is handled in different ways by different browsers. Some elements can be handled inline, some requires a plug-in and some requires an ActiveX control.

4.2.1 Audio Support in Browsers

Sound is a vital element of true multimedia Web pages. Sound can be stored in many different formats.

The MIDI Format

The MIDI (Musical Instrument Digital Interface) is a format for sending music information between electronic music devices like synthesizers and PC sound cards. The MIDI format was developed in 1982 by the music industry. The MIDI format is very flexible and can be used for everything from very simple to real professional music making. MIDI files do not contain sampled sound, but a set of digital musical instructions (musical notes) that can be interpreted by your PC's sound card.

The RealAudio Format

The RealAudio format was developed for the Internet by Real Media. The format also supports video. The format allows streaming of audio (on-line music, Internet radio) with low bandwidths. Because of the low bandwidth priority, quality is often reduced. Sounds stored in the RealAudio format have the extension .rm or .ram.

The AU Format

The AU format is supported by many different software systems over a large range of platforms. Sounds stored in the AU format have the extension .au.

The AIFF Format

The AIFF (Audio Interchange File Format) was developed by Apple. AIFF files are not cross-platform and the format is not supported by all web browsers. Sounds stored in the AIFF format have the extension .aif or .aiff.

The SND Format

The SND (Sound) was developed by Apple. SND files are not cross-platform and the format is not supported by all web browsers. Sounds stored in the SND format have the extension .snd.

The WAVE Format

The WAVE (waveform) format is developed by IBM and Microsoft. It is supported by all computers running Windows, and by all the most popular web browsers. Sounds stored in the WAVE format have the extension .wav.

The MP3 Format (MPEG)

MP3 files are actually MPEG files. But the MPEG format was originally developed for video by the Moving Pictures Experts Group. We can say that MP3 files are the sound part of the MPEG video format. MP3 is one of the most popular sound formats for music recording. The MP3 encoding system combines good compression (small files) with high quality. Expect all your future software systems to support it. Sounds stored in the MP3 format have the extension .mp3, or .mpga (for MPG Audio).

What Format To Use?

The WAVE format is one of the most popular sound format on the Internet, and it is supported by all popular browsers. If you want recorded sound (music or speech) to be available to all your visitors, you should use the WAVE format. The MP3 format is the new and upcoming format for recorded music. If your website is about recorded music, the MP3 format is the choice of the future.

4.2.2 Video Support in Browser

Like audio files, video files can be compressed to reduce the amount of data being sent. Because of the degree of compression required by video, most video codecs use a lossy approach that involves a trade-off between picture/sound quality and file size, with larger file sizes obviously resulting in longer download times. Video can be stored in many different formats.

The AVI Format

The AVI (Audio Video Interleave) format was developed by Microsoft. The AVI format is supported by all computers running Windows, and by all the most popular web browsers. It is a very common format on the Internet, but not always possible to play on non-Windows computers. Videos stored in the AVI format have the extension .avi.

The Windows Media Format

The Windows Media format is developed by Microsoft. Windows Media is a common format on the Internet, but Windows Media movies cannot be played on non-Windows computer without an extra (free) component installed. Some later Windows Media movies cannot play at all on non-Windows computers because no player is available. Videos stored in the Windows Media format have the extension .wmv.

The MPEG Format

The MPEG (Moving Pictures Expert Group) format is the most popular format on the Internet. It is cross-platform, and supported by all the most popular web browsers.Videos stored in the MPEG format have the extension .mpg or .mpeg.

The QuickTime Format

The QuickTime format is developed by Apple. QuickTime is a common format on the Internet, but QuickTime movies cannot be played on a Windows computer without an extra (free) component installed. Videos stored in the QuickTime format have the extension .mov.

The RealVideo Format

The RealVideo format was developed for the Internet by Real Media. The format allows streaming of video (on-line video, Internet TV) with low bandwidths. Because of the low bandwidth priority, quality is often reduced. Videos stored in the RealVideo format have the extension .rm or .ram.

The Shockwave (Flash) Format

The Shockwave format was developed by Macromedia. The Shockwave format requires an extra component to play. This component comes preinstalled with the latest versions of Netscape and Internet Explorer. Videos stored in the Shockwave format have the extension .swf.

4.2.3 Other Binary Formats in HTML

PDF Format

The term "PDF" stands for "Portable Document Format". The key word is portable, intended to combine the qualities of authenticity, reliability and ease of use together into a single packaged concept.

To be truly portable, an authentic electronic document would have to appear exactly the same way on any computer at any time, at no cost to the user. It will deliver the exact same results in print or on-screen with near-total reliability.

The difference between PDF and formats used for writing (Word, Excel, Power Point, Quark, HTML, etc) is profound. Properly made, PDF files are not subject to the vagaries of other formats. PDFs are not readily editable - and editing may be explicitly prohibited. A precise snapshot, a PDF file is created at a specific date and time, and in a specific way. You can trust a PDF like you can trust a fax. You can't say that about a Word file!

Adobe Systems invented PDF technology in the early 1990s to smooth the process of moving text and graphics from publishers to printing-presses. At the time, expectations were modest, but no longer. PDF turned out to be the very essence of paper, brought to life in a computer. In creating PDF, Adobe had almost unwittingly invented nothing less than a bridge between the paper and computer worlds.

4.3 Style Sheets

STYLE SHEET IS A COLLECTION OF FORMATTING STYLES, WHICH CAN BE APPLIED TO A WEB PAGE.

Benefits of the Style Sheets

Separation of style and content has many benefits:

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

Maintainability. Holding all the presentation styles in one file significantly reduces maintenance time, and reduces opportunity for human errors by reducing the maintenance tasks.

Accessibility. Sites that use CSS with either XHTML or HTML are easier to draw so that they appear extremely similar in different browsers

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

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

Portability. The suspension of presentational details until the time of presentation means that a document can be easily re-purposed for an entirely different presentation medium with merely the application of a new stylesheet already prepared for the new medium and consistent with elemental or structural vocabulary of the semantic document.

Disadvantages of the Style Sheets

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

Lack of semantic vocabulary. HTML offers a rich, but limited vocabulary of semantic elements (for example paragraph, quote, emphasis). The migration of HTML to the extensible XHTML may eventually speed the proliferation of richer semantic vocabularies to apply generalized styles to.

Complex Layouts. One of the practical problems is the lack of proper support for style languages in major browsers. Typical web page layouts call for some tabular presentation of the major parts of the page such as menu navigation columns and header bars, navigation tabs, and so on. However, deficient support for CSS and XSL in major browsers forces authors to code these tables within their content rather than applying a tabular style to the content from the accompanying stylesheet.

Narrow Adoption without the Parsing & Generation Tools. While the style specifications are quite mature and still maturing, the software tools have been slow to adapt. Most of the major web development tools still embrace a mixed presentation-content model. So authors and designers looking for GUI based tools for their work find it difficult to follow the semantic web method. In addition to GUI tools, shared repositories for generalized stylesheets would probably aid adoption of these methods.

Style Rule

A STYLE RULE IS A SET OF HTML TAGS SPECIFYING THE FORMATTING ELEMENTS.

Style rules can be applied to selected contents of a web page.

A style rule can basically be split into two parts:

(a) Selector: A selector is a string that identifies what elements the corresponding rule applies to and is the first part of the rule.

(b) Declaration: This part of the rule is enclosed within curly brackets. A declaration has two sections separated by a colon. The section before the colon tells the property, and the section after the colon is the value of that property.

The syntax of a style rule is as follows:

Selector (property : value)

Where,

Selector = Any HTML tag

Property = Attribute like font color, font size, etc.

Value = Setting for the attribute

e.g. H1 { color : blue }

H1 is the selector, color : blue is the declaration color is the property and blue is the value.

There are four ways of incorporating style sheets in HTML document, which are:

- Inline Style Sheet
- Internal style sheet
- Linking to an external style sheet
- Importing a style sheet

4.3.1 Inline Style Sheet

Inline style declaration is the most basic style rule, which can be applied to individual elements in the web page, Inline styles are implemented by using style attributes with the HTML tags.

The syntax is :

<HTML TAG STYLE = “PROPERTY : VALUE”>

e.g.

<P STYLE = { COLOR : OLIVE }>

Example: The style rule is applied to H1 tag by specifying it in the STYLE attribute.

```
<HTML>
<BODY>
<H1 STYLE = “Color : limegreen”> This is a style applied to an H1 element </H1>
<H1> This is the default display of an H1 element </H1>
</BODY>
</HTML>
```

The color property sets the color of the first H1 element to limegreen color and the second one remains the default since no style attribute is applied.

4.3.2 Internal Style Sheet

More than one style rule can be grouped by using <STYLE>…….</STYLE> tag pair unlike of applying it individually in inline style. Each of these tags when used in the BODY of the HTML code will apply the style rules.

The syntax is:

```
<HTML>
    <HEAD>
        <STYLE>
        Style Rules
        </STYLE>
    </HEAD>
</HTML>
<BODY>
    ……
    ……
    ……
</BODY>
</HTML>
```

Example,

```
<HTML>
<HEAD>
    <STYLE>
    H1 { font – family : Arial }
    H1 { Color : limegreen }
    </STYLE>
</HEAD>
<BODY>
    <H1> This is the H1 element </H1>
```

<H2> This is the H2 element </H2>

……

</BODY>

</HTML>

4.3.2.1 Selectors

These are the easiest to use. A SIMPLE SELECTOR DESCRIBES AN ELEMENT IRRESPECTIVE OF ITS POSITION IN THE DOCUMENT STRUCTURE. The H1 heading identifier is a typical example of simple selector.

H1 { Color : blue }

Simple selectors can be further classified into:

(a) HTML Selector

These selectors use the names of HTML elements without brackets. So the HTML <P> tag becomes P.

Example: In this example, while defining the style, the P element does not have brackets.

<HTML>

<HEAD>

<STYLE>

P { font – style : italic ; font – weight : bold ; color : limegreen ; font – size : 12 pt ; line – height : 16 pt }

</STYLE>

</HEAD>

<BODY>

<P> these selectors use the names of HTML elements. The only difference is that the brackets are removed. </P>

</BODY></HTML>

(b) Class Selector

The class selector gives authors the ability to apply styles to specific parts of a document and do not necessarily to the whole document.

The syntax is:

<STYLE>

Class selector

. Class Name Class selector { Property : Value }

</STYLE>

<BODY>

<P Class = “Class Name”>

Class attribute

</BODY>

Class name can be any valid string of characters.

The class selector is preceded with a dot (.) called the flag character

Class selector can be applied to any of the HTML elements by using the class attribute.

Example:

<HTML>

<HEAD>

<STYLE>

. note { color : blue }

. syntax { color : red }

P. syntax { color : red }

P { font – size : large }

</STYLE>

</HEAD>

<BODY>

<P CLASS = “syntax”> The class selector is preceded with a dot (.) called the flag character, followed by the ‘Class name’.

<P CLASS = “note”> It is better to choose class name according to their purpose rather than a name that describes their color or style.

<H1 CLASS = “note”> The class attribute is used even to the heading tag.

</H1>

</BODY>

</HTML>

(c) ID Selector

Like class selection, ID selector is also used to apply style to the selected parts of text. In this style, each ID selector has a unique identifier. An ID selector is preceded by a hash (#) mark and to apply ID selector the ID attribute of an HTML element is used.

The syntax is:

<STYLE>
ID Selector name
IDSelectorName {Property : Value}
</STYLE>
<BODY>
<P ID = “IDSelectorName”>
ID Attribute
</BODY>

Example:

<HTML>
<HEAD>
<TITLE> ID SELECTOR </TITLE>
</HEAD>
<BODY>
<STYLE>
#Control { color : red }
</STYLE>
<H2 Id = “Control”> Fire is this color </H2>

<H2> The browser controlled display of an H2 heading </H2>

<P ID = “Control”> Applying an ID to a paragraph element. </P>

<P> This is paragraph that has no style applied </P>
</BODY></HTML>

(d) Contextual Selectors

Consider a situation where, some tags are under H1 that are italics. Now to change any of their color to red following code could be used:

```
H1 { color : red }
I { color : red }
```

But in the above code, all type italicized tags turn to red. This is because of the line 2 of the code.

To avoid such situation contextual selectors can be used. Contextual selectors can be used to combine number of simple selector separated by a space.

The above code now can be rewritten as:

```
H1  I  { COLOR : RED }
```

The concept of contextual selectors support inheritance

Example:

```
<HTML>
<HEAD>
        <TITLE> contextual selectors </TITLE>
        <STYLE>
        body {
                Color : Magenta;
                Background : white;
                Font – family : Arial;
                }
        ul {
                color : aqua
            }
</STYLE>
</HEAD>
</HTML>
```

4.3.3 Cascading Style Sheet

The above methods of modifying styles of elements are within the same page. To apply similar settings for all the pages in the website, all the style rules should be put into a style sheet File and then imported or linked with the HTML document. This method of linking or importing is called *CASCADING STYLE SHEETS (CSS).*

4.3.3.1 Linking to an External Style Sheet

For constructing a CSS, first style rules must be written in a document and saves in a separate file with extension of CSS. The syntax for linking to an external style sheet is:

```
<HTML>
<HEAD>
<LINK REL = "STYLESHEET" HREF = "Dictionary path where the style sheet is saved">
</HEAD>
<BODY>
        ……
        ……
</BODY>
</HTML>
```

Where REL = STYLESHEET – Relation to the external document in a style sheet.

For example, consider a style sheet definition which specifies two styles:

- a global style applies to all <H2> element (green, arial font family, normal size)
- a generic style class named warning (red, bold, italic) which will apply to any element which uses that class.

Style 1. CSS has the following code:

```
<STYLE>
        H2 { COLOR : limegreen }
            font – family : "Arial";
            font – size : normal
             }
        .Warning { color : red;
```

```
    font – weight : bold;
  font –style : italic
 }
 </STYLE>
```

HTML file:

```
<HTML>
<HEAD>
<TITLE> Changing the rules </TITLE>
<link rel=”stylesheet” href =”…”>
</HEAD>
<H2> Changing the rules is fun </H2>
        <BR>
        <P class = “warning”> Changing the rules may not be such fun
<H2> The H2 element again </H2>
</HTML>
TYPE = “text / css”
```

4.3.3.2 Importing a Style Sheet

Importing a style sheet, automatically pulls, the style rules into the document for use. Once imported, changes made to the style sheet will not be reflected in the web page into which it has been imported. This problem can be avoided by linking the style sheet to the main document rather than importing it.

The syntax is:

```
<HTML>
<HEAD>
<STYLE TYPE = “TEXT / CSS”>
@ IMPORT URL (The Path) ;
</STYLE>
</HEAD>
```

<BODY>

……

……

</BODY></HTML>

4.4 CSS Basic Interactivity

Statements

A CSS style sheet is composed from a list of statements. A statement is either an at-rule or a rule set. The following example has two statements; the first is an at-rule that is delimited by the semicolon at the end of the first line, and the second is a rule set that is delimited by the closing curly brace, }:

```
import url(base.css);
h2 {
color: #666;
font-weight: bold;
}
```

At-rules

An at-rule is an instruction or directive to the CSS parser. It starts with an at-keyword: an @ character followed by an identifier. An at-rule can comprise a block delimited by curly braces, {…}, or text terminated by a semicolon, ;. An at-rule's syntax will dictate whether it needs a block or text—see CSS At-rules for more information.

Parentheses, brackets, and braces must appear as matching pairs and can be nested within the at-rule. Single and double quotes must also appear in matching pairs.

Rule Sets

A rule set (also called a rule) comprises a selector followed by a declaration block; the rule set applies the declarations listed in the declaration block to all elements matched by the selector.

Here's an example of a rule set:

```
h2 {
color: #666;
```

```
font-weight: bold;
}
```

Selectors

A selector comprises every part of a rule set up to—but not including—the left curly brace {. A selector is a pattern, and the declarations within the block that follows the selector are applied to all the elements that match this pattern. In the following example rule set, the selector is h2:

```
h2 {
color: #666;
font-weight: bold;
}
```

Declaration Blocks

Declaration blocks begin with a left curly brace, {, and end with a right curly brace, }. They contain zero or more declarations separated by semicolons:

```
h2 {
color: #666;
}
```

A declaration block is always preceded by a selector. We can combine multiple rules that have the same selector into a single rule. Consider these rules:

```
h2 {
color: #666;
}
h2 {
font-weight: bold;
}
```

They’re equivalent to the rule below:

```
h2 {
color: #666;
font-weight: bold;
}
```

CSS Comments

In CSS, a comment starts with /* and ends with */. Comments can span multiple lines, but may not be nested:

/* This is a single-line comment */

/* This is a comment that
spans multiple lines */

4.5 Positioning with Style sheets

Absolute Positioning

If you position an element (an image, a table, or whatever) absolutely on your page, it will appear at the exact pixel you specify. Say I wanted a graphic to appear 46 pixels from the top of the page and 80 pixels in from the right, I could do it. The CSS code you'll need to add into the image is

img {position: absolute; top: 46px; right: 80px; }

You just add in which method of positioning you're using at the start, and then push the image out from the sides it's going to be closest to. You can add the CSS directly into the tag using the style attribute (as shown in the introduction to stylesheets), or you can use classes and ids and put them into your stylesheet. It works the same way. The recommended method is to add classes for layout elements that will appear on every page, but put the code inline for once-off things.

Relative Positioning

An element whose position property has the value relative is first laid out just like a static element. The rendered box is then shifted vertically (according to the top or bottom property) and/or horizontally (according to the left or right property).

The properties top, right, bottom, and left can be used to specify by how much the rendered box will be shifted. A positive value means the box will be shifted away from that position, towards the opposite side. For instance, a left value of 20px shifts the box 20 pixels to the right of its original position. Applying a negative value to the opposite side will achieve the same effect: a right value of -20px will accomplish the same result as a left value of 20px.

The initial value for these properties is auto, which makes the computed value 0 (zero)—that is, no shift occurs.

Evidently, it's pointless to specify both left and right for the same element, because the position will be over-constrained. If the content direction is left to right, the left value is used, and right will be ignored. In a right-to-left direction, the right value "wins." If both top and bottom are specified, top will be used and bottom will be ignored.

Since it's only the rendered box that moves when we relatively position an element, this positioning scheme isn't useful for laying out columns of content. Relative positioning is commonly used when we need to shift a box a few pixels or so, although it can also be useful, in combination with negative margins on floated elements, for some more complex designs.

Fixed Positioning

Fixed positioning is a subcategory of absolute positioning. An element whose position property is set to fixed always has the viewport as its containing block. For continuous media, such as a computer screen, a fixed element won't move when the document is scrolled. For paged media, a fixed element will be repeated on every page.

Floating

A floated element is one whose float property has a value other than none. The element can be shifted to the left (using the value left) or to the right (using the value right); non-floated content will flow along the side opposite the specified float direction.

4.6 Style Sheet Properties

a) Font Properties.

(a) font-family – Denotes font

(b) font-size – Denotes the size of the text.

(c) font-style – Denotes the style of the text for example normal, bold, italic etc.

(d) font-weight – Denotes the weight or darkness of the font. This value ranges from 100 to 900 increments of 100.

b) Text Properties.

(a) Letter-spacing – Denotes the space between letters (in inches(in), centimeters (cm), millimeter (mm), or points (pt), etc.).
(b) Word-spacing – Denotes the space between words (in inches(in), centimeters (cm), millimeter (mm), or points (pt), etc.).
(c) Vertical-Align – Denotes the vertical positioning of the text and images, with respect to the baseline. The possible values include baseline, sub, top, text-top middle percentage values, etc.
(d) Text-Align – Specifies the alignment of the text. Possible values: center, justify, etc.
(e) Text-indent – Denotes margins. This property sets the indentation for text in the first line of block level element.
(f) Text-decorate – Denotes the text's decoration. The standard values for this property include blink, line – through over line underline etc.

c) Color and Background Properties.

(a) Color – Denotes the color property, used to set text color.
(b) Background-color – This property sets an element's background color.
(c) Background-image – Associates a background image with an element.
(d) Background-position – Specifies how a background image is positioned.

d) Box Properties.

Block style elements such as the <P> element can be considered as rectangular boxes on the screen. These properties include:

(a) Margin Properties – The margin values should be in length like 15 pt. etc. The individual margins for a block element can be set using margin-top, margin-rigid, margin-bottom, or margin-left.

(b) Border Properties:

- **Border-style –** Used to set the appearance of the borders. The values can include solid, double, groove, ridge, etc.

- **Border-width** – The width of the border is specified here. The individual border width can be set using border-top-width border-right-width, border-bottom-width, border-left-width etc.
- **Border-color** – Border may be assigned a color.

4.7 Overview of XML

The **Extensible Markup Language** (**XML**) is a general-purpose *specification* for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet.

Syntax of XML

XML is a generic framework for storing any amount of text or any data whose structure can be represented as a tree. The only requirement is that the text must be enclosed between a start tag and a corresponding end tag. For example,

<BOOK> This is a book… </BOOK>

< ? XML VERSION = “1.0” ENCODING = “UTF – 8”? >

This element states what version of XML is in use. It may also contain information about character encoding.

The root element can be preceded by this optional XML declaration.

Comments can be placed anywhere in the tree,

< ! - - This is a comment - - >

The basic syntax of an element is:

<NAME ATTRIBUTE = “value”> text </NAME>

Tree structure of an XML document:

```
<ROOT>
    <CHILD>
        <SUBCHILD> … </SUBCHILD>
    </CHILD>
</ROOT>
```

For example,

<BOOKSTORE>

<BOOK CATEGORY = “children”>

<TITLE> Harry Potter </TITLE>

<AUTHOR> J.K. Rowling </AUTHOR>

<YEAR> 2005 </YEAR>

<PRICE> 29.99 </PRICE>

</BOOK>

<BOOK CATEGORY = “web”>

<TITLE> Learning XML </TITLE>

<AUTHOR> Erik T. Ray </AUTHOR>

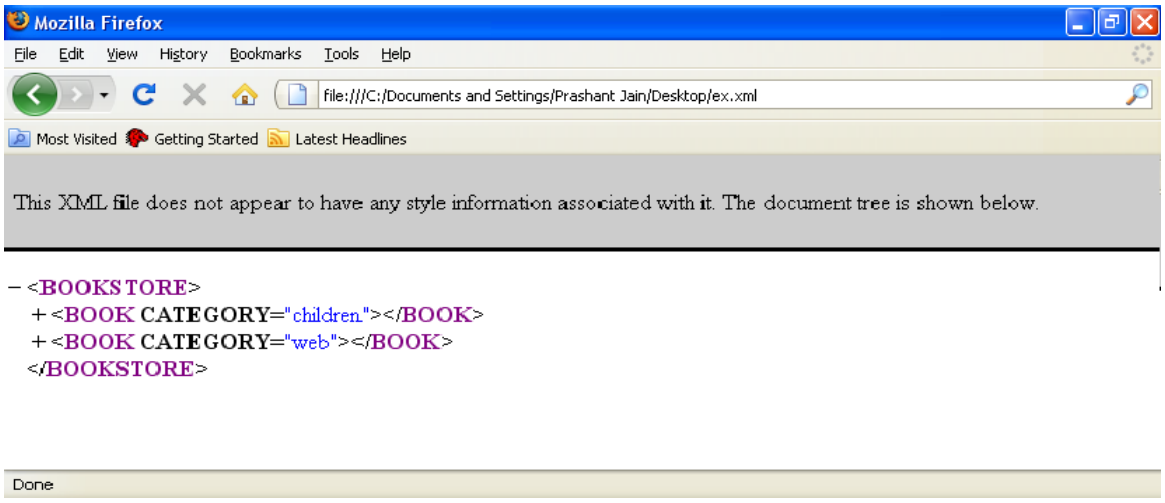
<YEAR> 2003 </YEAR>

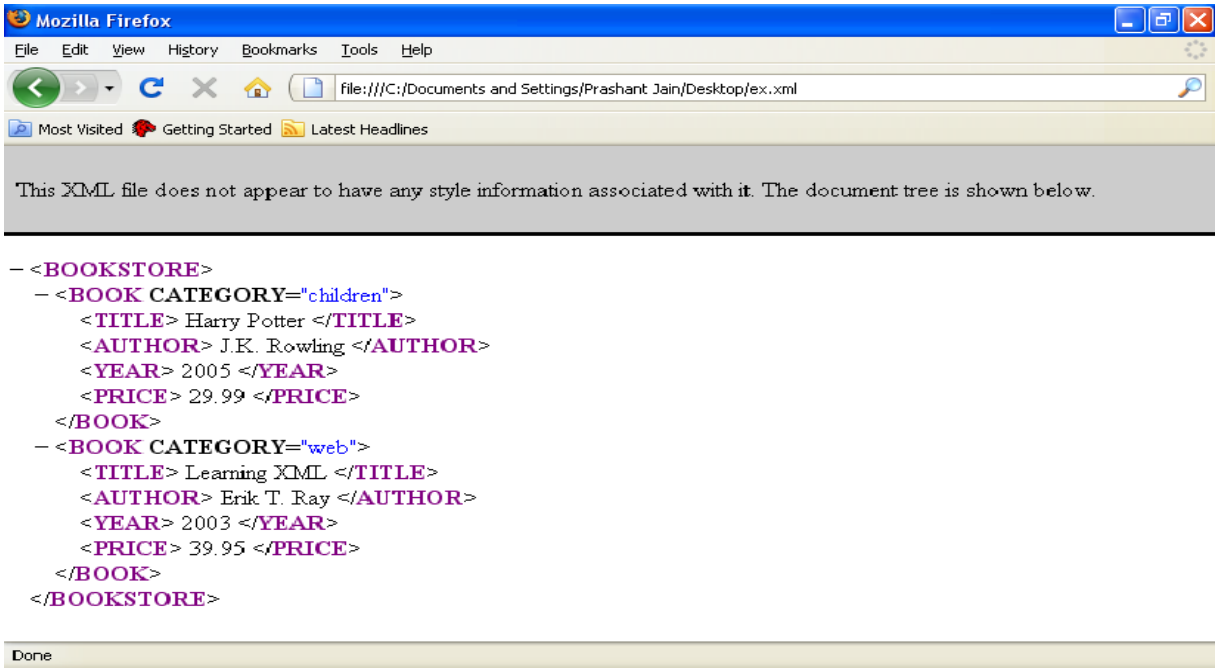
<PRICE> 39.95 </PRICE>

</BOOK>

</BOOKSTORE>

After creating the XML document as mentioned above, the page will be displayed as follows:





Advantages of XML

1. It is text – based.
2. It can represent common computer science data structures: records, lists, and trees.
3. Its self-documenting format describes structure and field names as well as specific values.
4. It is platform – independent.
5. It can be updated incrementally.

Disadvantages of XML

1. XML syntax is redundant or large relative to binary representations of similar data.
2. The redundancy may affect application efficiency through higher storage, transmission, and processing costs.
3. Expressing non – hierarchical node relations requires extra effort.

Why do we need XML?

Data-exchange

1).XML is used to aid the exchange of data. It makes it possible to define data in a clear way.

2). Both the sending and the receiving party will use XML to understand the kind of data that's been sent. By using XML everybody knows that the same interpretation of the data is used.

Replacement for EDI

1).EDI (Electronic Data Interchange) has been for several years the way to exchange data between businesses.

2).EDI is expensive, it uses a dedicated communication infrastructure. And the definitions used are far from flexible.

3).XML is a good replacement for EDI. It uses the Internet for the data exchange. And it's very flexible.

More possibilities

1).XML makes communication easy. It's a great tool for transactions between businesses.

2).But it has much more possibilities. You can define other languages with XML. A good example is WML (Wireless Markup Language), the language used in WAPcommunications. WML is just an XML dialect.

What it can do

With XML you can :

- Define data structures
- Make these structures platform independent
- Process XML defined data automatically
- Define your own tags

With XML you cannot

- Define how your data is shown. To show data, you need other techniques.

Define your own tags

In XML, you define your own tags.

If you need a tag <TUTORIAL> or <STOCKRATE>, that's no problem.

DTD or Schema

If you want to use a tag, you'll have to define it's meaning. This definition is stored in a DTD (Document Type Definition). You can define your own DTD or use an existing one. Defining a DTD actually means defining a XML language. An alternative for a DTD is Schema.

Showing the results

Often it's not necessary to display the data in a XML document. It's for instance possible to store the data in a database right away. If you want to show the data, you can. XML itself is not capable of doing so. But XML documents can be made visible with the aid of a language that defines the presentation. XSL (eXtensible Stylesheet Language) is created for this purpose. But the presentation can also be defined with CSS (Cascading Style Sheets).

Tags

XML tags are created like HTML tags. There's a start tag and a closing tag.

<TAG>content</TAG>

The closing tag uses a slash after the opening bracket, just like in HTML.

The text between the brackets is called an element.

Syntax

The following rules are used for using XML tags:

1).Tags are case sensitive. The tag <TRAVEL> differs from the tags <Travel> and <travel>.

2).Starting tags always need a closing tag.

3).All tags must be nested properly.

4).Comments can be used like in HTML:

5).Between the starting tag and the end tag XML expects the content.

<amount>135</amount> is a valid tag for an element amount that has the content 135.

Empty tags

Besides a starting tag and a closing tag, you can use an empty tag. An empty tag does not have a closing tag. The syntax differs from HTML:

Empty Tag : <TAG/>

Elements and children

With XML tags you define the type of data. But often data is more complex. It can consist of several parts. To describe the element car you can define the tags <car>mercedes</car>. This model might look like this:

<car>
<brand>volvo</brand>
<type>v40</type>
<color>green</color> </car>

Besides the element car three other elements are used: brand, type and color. Brand, type and color are sub-elements of the element car. In the XML-code the tags of the sub-elements are enclosed within the tags of the element car. Sub-elements are also called children.

4.7.1 The Future of XML

The future of XML is still unclear because of conflicting views of XML users. Some say that the future is bright and holds promise. While others say that it is time to take a break from the continuous increase in the volume of specifications.

In the past five years, there have been substantial accomplishments in XML. XML has made it possible to manage large quantities of information which don't fit in relational database tables, and to share labeled structured information without sharing a common Application Program Interface (API). XML has also simplified information exchange across language barriers.

But as a result of these accomplishments, XML is no longer simple. It now consists of a growing collection of complex connected and disconnected specifications. As a result , usability has suffered. This is because it takes longer to develop XML tools. These users are now rooting for something simpler. They argue that even though specifications have increased, there is no clear improvement in quality. They think in might be better to let things be, or even to look for alternate approaches beyond XML. This will make XML easier to use in the future. Otherwise it will cause instability with further increase in specifications.

The other side paints a completely different picture. They are ready for further progress in XML. There have been discussions for a new version, XML 2.0. This version has been proposed to contain the following characteristics:

- § Elimination of DTDS
- § Integration of namespace
- § XML Base and XML Information Set into the base standard

Research is also being carried out into the properties and use cases for binary encoding of the XML information set.

Future of XML Applications

The future of XML application lies with the Web and Web Publishing. Web applications are no longer traditional. Browsers are now integrating games, word processors and more. XML is based in Web Publishing, so the future of XML is seen to grow as well.

4.8 XML with HTML and SGML

First you should know that SGML (Standard Generalized Markup Language) is the basis for both HTML and XML. SGML is an international standard (ISO 8879) that was published in 1986.

Second, you need to know that XHTML is XML. "XHTML 1.0 is a reformulation of HTML 4.01 in XML, and combines the strength of HTML 4 with the power of XML."

Thirdly, XML is NOT a language, it is rules to create an XML based language. Thus, XHTML 1.0 uses the tags of HTML 4.01 but follows the rules of XML.

The Document

A typical document is made up of three layers:

- structure
- Content
- Style

Structure

Structure would be the documents title, author, paragraphs, topics, chapters, head, body etc.

Content

Content is the actual information that composes a title, author, paragraphs etc.

Style

Style is how the content within the structural elements are displayed such as font color, type and size, text alignment etc.

Markup

HTML, SGML, and XML all markup content using tags. The difference is that SGML and

XML mainly deal with the relationship between content and structure, the structural tags that markup the content are not predefined (you can make up your own language), and style is kept TOTALLY separate; HTML on the other hand, is a mix of content marked up with both structural and stylistic tags. HTML tags are predefined by the HTML language.

By mixing structure, content and style you limit yourself to one form of presentation and in HTML's case that would be in a limited group of browsers for the World Wide Web.

By separating structure and content from style, you can take one file and present it in multiple forms. XML can be transformed to HTML/XHTML and displayed on the Web, or the information can be transformed and published to paper, and the data can be read by any XML aware browser or application.

SGML (Standard Generalized Markup Language)

Historically, Electronic publishing applications such as Microsoft Word, Adobe PageMaker or QuarkXpress, "marked up" documents in a proprietary format that was only recognized by that particular application. The document markup for both structure and style was mixed in with the content and was published to only one media, the printed page.

These programs and their proprietary markup had no capability to define the appearance of the information for any other media besides paper, and really did not describe very well the actual content of the document beyond paragraphs, headings and titles. The file format could not be read or exchanged with other programs, it was useful only within the application that created it.

Because SGML is a nonproprietary international standard it allows you to create documents that are independent of any specific hardware or software. The document structure (what elements are used and their relationship to each other) is described in a file called the DTD (Document Type Definition). The DTD defines the relationships between a document's elements creating a consistent, logical structure for each document.

SGML is good for handling large-scale, long-term information management needs and has been around for more than a decade as the language of defense contractors and the electronic publishing industry. Because SGML is very large, powerful, and complex it is hard to learn and understand and is not well suited for the Web environment.

XML (Extensible Markup Language)

XML is a "restricted form of SGML" which removes some of the complexity of SGML. XML like SGML, retains the flexibility of describing customized markup languages with a user-defined document structure (DTD) in a non-proprietary file format for both storage and exchange of text and data both on and off the Web.

As mentioned before, XML separates structure and content from style and the structural markup tags can actually describe the content because they can be customized for each XML based markup language. A good example of this is the Math Markup Language (MathML) which is an XML application for describing mathematical notation and capturing both its structure and content.

Until MathML, the ability to communicate mathematical expressions on the Web was limited to mainly displaying images (JPG or GIF) of the scientific notation or posting the document as a PDF file. MathML allows the information to be displayed on the Web, and makes it available for searching, indexing, or reuse in other applications.

HTML (Hypertext markup Language)

HTML is a single, predefined markup language that forces Web designers to use it's limiting and lax syntax and structure. The HTML standard was not designed with other platforms in mind, such as Web TV's, mobile phones or PDAs. The structural markup does little to describe the content beyond paragraph, list, title and heading.

XML breaks the restricting chains of HTML by allowing people to create their own markup languages for exchanging information. The tags can be descriptive of the content and authors decide how the document will be displayed using style sheets (CSS and XSL). Because of XML's consistent syntax and structure, documents can be transformed and published to multiple forms of media and content can be exchanged between other XML applications.

HTML was useful in the part it has played in the success of the Web but has been outgrown as the Web requires more robust, flexible languages to support it's expanding forms of communication and data exchange.

In Short

XML will never completely replace SGML because SGML is still considered better for long-time storage of complex documents. However, XML has already replaced HTML as

the recommended markup language for the Web with the creation of XHTML 1.0.
Even though XHTML has not made the HTML that currently exists on the Web obsolete, HTML 4.01 is the last version of HTML. XHTML (an XML application) is the foundation for a universally accessible, device independent Web.

4.9 Check your progress

1. MIDI stands for …….
2. AIFF stands for …….
3. AVI stands for ……..
4. "PDF" stands for ……...
5. A ……...is a string that identifies what elements the corresponding rule applies to and is the first part of the rule.
6. The class selector is preceded with a dot (.) called the …….
7. …….selectors support inheritance.
8. four types of Positioning with Style sheets are Absolute, relative, fixed and…….
9. ……..property Denotes margins. This property sets the indentation for text in the first line of block level element.
10. The …….is a general-purpose *specification* for creating custom markup languages.
11. …….is used to aid the exchange of data.
12. EDI stands for…….
13. SGML stands for……...

4.10 Summary

Multimedia is everything you can hear or see: texts, books, pictures, music, sounds, CDs, videos, DVDs, Records, Films, and more.
Multimedia comes in many different formats. On the Internet you will find many of these elements embedded in web pages, and today's web browsers have support for a number of multimedia formats.

STYLE SHEET IS A COLLECTION OF FORMATTING STYLES, WHICH CAN BE APPLIED TO A WEB PAGE.

A STYLE RULE IS A SET OF HTML TAGS SPECIFYING THE FORMATTING ELEMENTS.

A style rule can basically be split into two parts:

(c) **Selector:** A selector is a string that identifies what elements the corresponding rule applies to and is the first part of the rule.

(d) **Declaration:** This part of the rule is enclosed within curly brackets. A declaration has two sections separated by a colon. The section before the colon tells the property, and the section after the colon is the value of that property.

There are four ways of incorporating style sheets in HTML document, which are:

- Inline Style Sheet
- Internal style sheet
- Linking to an external style sheet
- Importing a style sheet

To apply similar settings for all the pages in the website, all the style rules should be put into a style sheet File and then imported or linked with the HTML document. This method of linking or importing is called *CASCADING STYLE SHEETS (CSS).*

For constructing a CSS, first style rules must be written in a document and saves in a separate file with extension of CSS. Importing a style sheet, automatically pulls, the style rules into the document for use. Once imported, changes made to the style sheet will not be reflected in the web page into which it has been imported.

The **Extensible Markup Language** (**XML**) is a general-purpose *specification* for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet.

XML will never completely replace SGML because SGML is still considered better for long-time storage of complex documents. However, XML has already replaced HTML as the recommended markup language for the Web with the creation of XHTML 1.0.
Even though XHTML has not made the HTML that currently exists on the Web obsolete, HTML 4.01 is the last version of HTML. XHTML (an XML application) is the foundation for a universally accessible, device independent Web.

4.11 Self-Assessment Test

Q.1 Explain various types of Audio formats supported by HTML.
Q.2. Briefly describe video and other binary formats supported by HTML.
Q.3 Describe selectors in Internal HTML Style sheets.
Q.4 Explain CSS and its properties.
Q.5. Compare XML with HTML and SGML.
Q.6 Explain various types of positioning in style sheets.
Q.7 How will you create document with XML? Give an example.

4.12 Answers to check your progress

1. Musical Instrument Digital Interface
2. Audio Interchange File Format
3. Audio Video Interleave
4. "Portable Document Format"
5. Selector
6. flag character
7. contextual
8. floating
9. Text-indent
10. Extensible Markup Language (XML)
11. XML
12. Electronic Data Interchange
13. Standard Generalized Markup Language

4.13 References / Suggested Readings

1. Thomas A Powell, HTML-The Complete Reference,Tata McGraw Hill.
2. Scott Guelich, Shishir Gundavaram, Gunther Birzniek; CGI Programming with Perl 2/e. O'Reilly.
3. Doug Tidwell, James Snell, Pavel Kulchenko; Programming Web Services with SOAP, O'Reilly.
4. Pardi, XML in Action, Web Technology, PHI.
5. Yong, XML Step by Step, PHI.
6. Aaron Weiss, Rebecca Taply, Kim Daniels, Stuven Mulder, Jeff Kaneshki, Web Authoring Desk Reference, Techmedia Publications.
7. HTML The complete Reference, TMH

SUBJECT: WEB DESIGNING	
COURSE CODE: MCA-12	**AUTHOR: DR. SUDHANSHU GAUR**
LESSON NO. 5	
Client-Side Programming	

STRUCTURE

5.0 LEARNING OBJECTIVE

After going through this unit, you will be able to:

- learn about the concept of client-side programming.
- describe the JavaScript Object Model.
- learn about various pop-up boxes in JavaScript.
- Learn about form handling in JavaScript.
- Learn about Cookies.

5.1 INTRODUCTION

This chapter briefly describe the concept of Client-Side Programming. We will have a look at the JavaScript language and its use in client-side programming. We will discuss about the JavaScript Object Model and various objects in JavaScript.

We will create programs for creating various types of popup boxes in JavaScript. We will have a look at the concept of form handling in JavaScript. In this we will discuss about cookies and how we will create cookies.

5.2 Introduction to JavaScript

JavaScript was designed to add interactivity to HTML pages.
It is a scripting language.
A scripting language is a lightweight programming language.
A JavaScript consists of lines of executable computer code.
It is usually embedded directly into HTML pages.
It is an interpreted language (means that scripts execute without preliminary compilation).
Everyone can use JavaScript without purchasing a license.

What a JavaScript can do?

It gives HTML designers a programming tool.
It can put dynamic text into an HTML page.

It can react to events.

It can read and write HTML elements.

It can be used to validate data.

It can be used to detect the visitor's browser.

It can be used to create cookies.

How to put?

<HTML>.

<BODY>

<SCRIPT TYPE="text/javascript">

Document.write("Hello World !")

</SCRIPT>

</BODY>

</HTML>

Objects, Properties and Methods in JavaScript

Each object has certain *properties,* and *methods* associated with it. Properties are things that describe the object, and they include sub – objects. For example, is a person is considered to be an object, then hair color, height, etc. are its properties or sub – objects.

Methods are things that the object can do or things that can be done to the object. One method associated with the document object is write(). For example, the document.write() method writes HTML to a Web page. The argument to be passed to write() method is a string of text that will be written. For example, document.write("Hello World!").

5.3 JavaScript Object Model

There are several objects supported by JavaScript. Some of these are explained as follows:

5.3.1 Window Object

The Window object is the top level object in JavaScript, and contains in itself several other objects, such as "document", "history" etc.

Events

Events handlers supported by a Window object may be:

Window Object Properties

Property	Description
closed	Returns a Boolean value indicating whether a window has been closed or not
console	Returns a reference to the Console object, which provides methods for logging information to the browser's console (See Console object)
defaultStatus	Sets or returns the default text in the statusbar of a window
document	Returns the Document object for the window (See Document object)
frameElement	Returns the <iframe> element in which the current window is inserted
frames	Returns all <iframe> elements in the current window
history	Returns the History object for the window (See History object)
innerHeight	Returns the height of the window's content area (viewport) including scrollbars
innerWidth	Returns the width of a window's content area (viewport) including scrollbars
length	Returns the number of <iframe> elements in the current window
localStorage	Allows to save key/value pairs in a web browser. Stores the data with no expiration date
location	Returns the Location object for the window (See Location object)
name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window (See Navigator object)
opener	Returns a reference to the window that created the window

outerHeight	Returns the height of the browser window, including toolbars/scrollbars
outerWidth	Returns the width of the browser window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screen	Returns the Screen object for the window (See Screen object)
screenLeft	Returns the horizontal coordinate of the window relative to the screen
screenTop	Returns the vertical coordinate of the window relative to the screen
screenX	Returns the horizontal coordinate of the window relative to the screen
screenY	Returns the vertical coordinate of the window relative to the screen
sessionStorage	Allows to save key/value pairs in a web browser. Stores the data for one session
scrollX	An alias of pageXOffset
scrollY	An alias of pageYOffset
self	Returns the current window
status	Sets or returns the text in the statusbar of a window
top	Returns the topmost browser window

Window Object Methods

Method	Description
alert()	Displays an alert box with a message and an OK button
atob()	Decodes a base-64 encoded string
blur()	Removes focus from the current window
btoa()	Encodes a string in base-64
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
focus()	Sets focus to the current window
getComputedStyle()	Gets the current computed CSS styles applied to an element
getSelection()	Returns a Selection object representing the range of text selected by the user
matchMedia()	Returns a MediaQueryList object representing the specified CSS media query string
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window

prompt()	Displays a dialog box that prompts the visitor for input
requestAnimationFrame()	Requests the browser to call a function to update an animation before the next repaint
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	Deprecated. This method has been replaced by the scrollTo() method.
scrollBy()	Scrolls the document by the specified number of pixels
scrollTo()	Scrolls the document to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds
stop()	Stops the window from loading

Program for Closed Property :

```
<html>
<head>
<script>
var myWindow;
function openWin() {
   myWindow = window.open("", "myWindow", "width=400, height=200");
}
function closeWin() {
   if (myWindow) {
      myWindow.close();
   }
}
```

```
function checkWin() {
   if (!myWindow) {
      document.getElementById("msg").innerHTML = "'myWindow' has never been opened!";
   } else {
      if (myWindow.closed) {
         document.getElementById("msg").innerHTML = "'myWindow' has been closed!";
      } else {
         document.getElementById("msg").innerHTML = "'myWindow' has not been closed!";
      }
   }
}
</script>
</head>
<body>
<button onclick="openWin()">Open "myWindow"</button>
<button onclick="closeWin()">Close "myWindow"</button>
<br><br>
<button onclick="checkWin()">Has "myWindow" been closed?</button>
<br><br>
<div id="msg"></div>
</body>
</html>
```

Output :

Open "myWindow" **Close "myWindow"**

Has "myWindow" been closed? ----------When Click on this Button

This Msg is printed -------'myWindow' has never been opened!

5.3.2 Frame Object

The frame object is a browser object of JavaScript used for accessing HTML frames. The user can use frames array to access all frames within a window. Using the indexing concept, users can access the frames array.

- The frames array index always starts with zero and not 1.
- The frame object is actually a child of the window object. These objects are created automatically by the browser and help users to control loading and accessing of frames.
- The properties and methods of frame object are similar to that of Window object in JavaScript.
- The frame object does not support close() method that is supported by window object.
- Using the <FRAMESET> document creates frame objects and each frame created is thus a property of window object.

Properties of frame object:

- frames
- name
- length
- parent
- self

frames:

The frames property of frame object denotes a collection or array of frames in a window and also in a frame set.

self:

As the name implies, the self property of frames object denotes the current frame. Using self property, the user can access properties of the current frame window.

name:

The name property of frame object denotes the name of the frame. The method of denoting the name attribute is performed by using the name attribute of the <frame>tag.

length:

The frames array has all the frames present within a window and the length property of the frame object denotes the length of the frames array or gives the number of frames present in a window or a frames array.

parent:

As the name implies, the parent property of frames object denotes the parent frame of the current frame.

Methods of frame object:

- blur()
- focus()
- setInterval()
- clearInterval()
- setTimeout(expression, milliseconds)
- clearTimeout(timeout)

blur():

blur() method of frame object removes focus from the object.

focus():

focus() method of frame object gives focus to the object.

setInterval():

setInterval() method of frame object is used to call a function of JavaScript or to evaluate an expression after the time interval specified in arguments has expired. The time interval in arguments is always specified in milliseconds.

clearInterval():

clearInterval method of frame object is used to cancel the corresponding definedsetInterval method. This is written by referencing the setInterval method using its ID or variable.

setTimeout(expression, milliseconds):

setTimeout method of frame object can be used to execute any function, or access any method or property after a specified time interval given to this method as argument.

clearTimeout():

clearTimeout method of frame object is used to clear a specified setTimeoutmethod. This is written by referencing the setTimeout method using its ID or variable.

5.3.3 Navigator Object

The Navigator object of JavaScript returns useful information about the visitor's browser and system.

The navigator object has three methods:

- javaEnabled specifies whether Java is enabled
- preference lets you use a signed script to get or set various user preferences (JavaScript 1.2 and later)
- taintEnabled specifies whether data tainting is enabled (JavaScript 1.1 only)

Properties and Methods supported by Navigator Object:

Property/Method	Description
appCodeName	Represents the code name of the browser
appName	Refers to the official browser name
appVersion	Refers to the version information of the browser
javaEnabled()	Function that tests to see that Java is supported in the browser
language	Refers to the language of the browser
mimeTypes	Refers to an array of MimeType objects that contains all the MIME types that the browser supports
platform	A string representing the platform on which the browser is running
plugins	Refers to an array of Plugin objects that contains all the plug-ins installed in the browser
plugins.refresh()	Checks for any newly installed plug-ins
preference()	Allows reading and setting of various user preferences in the browser

taintEnabled()	Tests to see whether data-tainting is enabled
userAgent	String that represents the user-agent header

5.3.4 Document Object

Document is the parent object of numerous other objects, such as "images", "forms" etc.

Properties of Document Object:

alinkColor	String, Specifies the ALINK attribute.
anchors[]	Array of anchor objects, contains an entry for each anchor in the document.
applets[]	Array of applet objects, Contains an entry for each applet in the document.
bgColor	String, Specifies the BGCOLOR attribute
cookie	String, Specifies a cookie.
domain	String, Specifies the domain name of the server that served a document.
embeds[]	Array, Contains an entry for each plug-in in the document.
fgColor	String, Specifies the TEXT attribute.
forms[]	Array of Form objects, Contains an entry for each form in the document
images[]	Array of Image objects, contains an entry for each image in the document.
lastModified	String, Specifies the date the document was last modified.
layers	Array of Layer objects, contains an entry for each layer within the document.

linkColor	String, Specifies the LINK attribute.
links[]	Array of Link objects, contains an entry for each link in the document.
referrer	String, Specifies the URL of the calling document.
title	String, Specifies the contents of the TITLE tag.
URL	String, Specifies the complete URL of a document.
vlinkColor	String, Specifies the VLINK attribute.

Methods of Document Object:

captureEvents()	Loads the previous URL in the history list.
close()	Closes an output stream and forces data to display.
getSelection()	Returns a string containing the text of the current selection
handleEvent()	Invokes the handler for the specified event.
open()	Opens a stream to collect the output of write or writeln methods.
releaseEvents()	Sets the window or document to release captured events of the specified type, sending the event to objects further along the event hierarchy.
routeEvent()	Passes a captured event along the normal event hierarchy.
write()	Writes one or more HTML expressions to a document in the specified window.
writeln()	Writes one or more HTML expressions to a document in the specified window and follows them with a newline character

5.4 Events

Events are actions that can be detected by JavaScript. Every element on a web page has certain events which can trigger JavaScript functions. For example, one can use the onClick event on a button element to indicate that a function will run when a user clicks on the button.

Examples of events:

- A mouse click
- A web page or an image loading
- Mouse over a hot spot on the web page.
- Selecting an input box in an HTML form
- Submitting an HTML form
- A keystroke

a) onLoad and onUnload

These events are triggered when the user enters or leaves the page. The onLoad event is often used to check the visitor's browser type and browser version, and load its proper version of the web page based on the information. Both the events are also often sued to deal with cookies that should be set when a user enters or leaves a page.

b) onFocus, onBlur and onChange

These events are often used in combination with validation of form fields.

c) onSubmit

The onSubmit event is used to validate ALL form fields before submitting it.

d) onMouseOver and onMouseOut

These events are often used to create "animated" buttons.

5.5 JavaScript Popup Boxes

JavaScript supported 3 types of popup boxes as explained below:

5.5.1 Alert Box

An alert box is often used if the programmer want to make sure information comes through to the user. When an alert box pops up, the user will have to click “OK” to proceed.

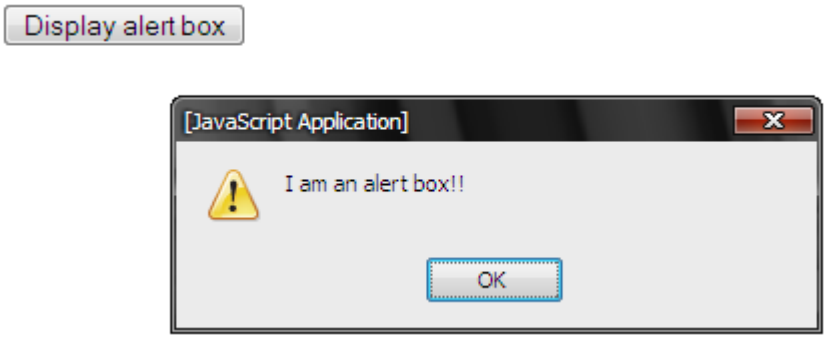
Example:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<SCRIPT TYPE="text/javascript">
<!--
function disp_alert()
{
alert("I am an alert box!!");
}
//-->
</SCRIPT>

<INPUT TYPE="button" onclick="disp_alert()" VALUE="Display alert box" />

</BODY>
</HTML>
```

Output:



5.5.2 Confirm Box

A confirm box is often used if the programmer want the user to verify or accept something. When a confirm box pops up, the user will have to click either “OK” or “Cancel” to proceed. If the user clicks “Cancel”, the box returns false.

Example:

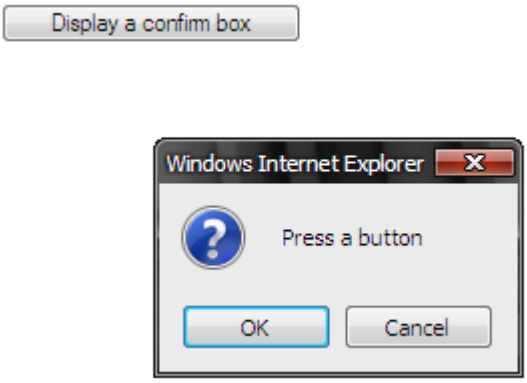
```
<HTML>
<HEAD>
</HEAD>

<BODY>
<SCRIPT TYPE="text/javascript">
<!--
function disp_confirm()
{
var r=confirm("Press a button");
if (r==true)
{
document.write("You pressed OK!");
}
else
{
document.write("You pressed Cancel!");
 }
 } //-->
</SCRIPT>

<INPUT TYPE="button" onclick="disp_confirm()" VALUE="Display a confirm box" />

</BODY></HTML>
```

Output:



5.5.3 Prompt Box

A prompt box is often used if the programmer want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either “OK” or “Cancel” to proceed after entering an input value. If the user clicks “OK”, the box returns the input value. If the user clicks “Cancel” the box returns null.

Example:

```
<HTML>
<HEAD>
</HEAD>

<BODY>
<SCRIPT TYPE="text/javascript">
<!--
function disp_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
//-->
</SCRIPT>
```

<INPUT TYPE="button" onclick="disp_prompt()" VALUE="Display a prompt box" />

</BODY>
</HTML>

Output:

Hello Harry Potter! How are you today?

5.6 Form Handling in JavaScript

JavaScript can be embedded into the HTML document by using <SCRIPT> tag to provide dynamic feature to various forms. For example, consider the following user – login form to verify the user name and password before redirecting to an authenticated Web page:

Redirect.html

<HTML>
<HEAD>
<TITLE>Login Form</TITLE>
</HEAD>

<BODY>
<SCRIPT LANGUAGE = "JavaScript">
<!--
function passuser(form)
{

```
if(form.id.value.length>=5)
        {
                if(form.id.value=="student")
                {
                        if(form.passwd.value=="college")
                        {
                                location = "next.html";
                        }
                        else
                        {
                                alert("Invalid Password !!");
                        }
                }
                else
                {
                        alert ("Invalid UserName!!");
                }
}
else
{
        alert(“Username should be more than 4 characters !!”);
 }
}
//-->
</SCRIPT>

<FORM>
<PRE>
<P>Username:          <INPUT TYPE = "text" NAME = "id"></P>
<P>Password: <INPUT TYPE = "password" NAME = "passwd"></P>
<CENTER><INPUT TYPE = "button" VALUE = "login" onClick =
"passuser(this.form)">
</CENTER>
```

```
</PRE>
</FORM>
</BODY>
</HTML>
```

next.html

```
<HTML>
<HEAD>
<TITLE>WELCOME</TITLE>
</HEAD>
<BODY BGCOLOR = "AQUA">
<P><CENTER>
"Congratulations!!! You have logged in successfully!!!"
</CENTER></P>
</BODY>
</HTML>
```

5.7 Cookies

A cookie is used to identify a user.

"A COOKIE IS A SMALL FILE THAT THE SERVER EMBEDS ON THE USER'S COMPUTER. EACH TIME THE SAME COMPUTER REQUESTS A PAGE WITH A BROWSER, IT WILL SEND THE COOKIE TOO."

5.7.1 Creating Cookies

The "Response.Cookies" command is used to create cookies. For example, to create a cookies named "firstname" and assigning the value "Ashwin" to it, following command may be used:

```
< %
Response . Cookies("firstname") = "Ashwin"
% >
```

It is also possible to assign properties to a cookie, like setting a date when the cookie should expire:

< %

Response . Cookies(“firstname”) = “Ashwin”

Response . Cookies(“firstname”) . Expires = # May 10, 2002 #

% >

5.7.2 Retrieve a Cookie Value

The “Request . Cookies” command is used to retrieve a cookie value. For example,

< %

Fname = Request . Cookies(“firstname”)

Response . write(“Firstname = ” & fname)

% >

Output:

Firstname = Ashwin

5.7.3 A Cookie With Keys

If a cookie contains a collection of multiple values, this is said as the cookie has keys. Consider the example to create a cookie collection named “user”. The “user” cookie has keys that contains information about a user:

< %

Response . Cookies(“user”) (“firstname”) = “ADITYA VARDHAN”

Response . Cookies(“user”) (“lastname”) = “ANUBHAV AGARWAL”

Response . Cookies(“user”) (“country”) = “INDIA”

Response . Cookies(“user”) (“age”) = “22”

% >

5.7.4 Read all Cookies

Consider the following code:

< %

Response . Cookies(“firstname”) = “Ashwin”

Response . Cookies(“user”) (“firstname”) = “ADITYA VARDHAN”

```
Response . Cookies(“user”) (“lastname”) = “ANUBHAV AGARWAL”
Response . Cookies(“user”) (“country”) = “INDIA”
Response . Cookies(“user”) (“age”) = “22”
% >
```

Assuming the server sent all the cookies above to a user. To read the cookies the user can use the following code:

```
<html>
<body>
< %
Dim x,y
For each x in Request . Cookies
        Response.write(“<p>”)
if Request . Cookies (x) . HasKeys then
        for each y in Request . Cookies (x)
                response.write (x & “:” & y & “=” & Request . Cookies (x) (y))
        response.write (“<br />”)
next
else
        Response . Write (x & “=” & Request . Cookies (x) & “<br />”)
end if
response. Write (“</p>”)
next
% >
</body>
</html>
```

Output:

```
Firstname = Ashwin
User:firstname = ADITYA VARDHAN
User:lastname = ANUBHAV AGARWAL
User:country = INDIA
User:age = 22
```

5.8 Check Your Progress

1. …….was designed to add interactivity to HTML pages.
2. Each ……..has certain properties, and methods associated with it.
3. The ……..object is the top level object in JavaScript.
4. The …….. object is a browser object of JavaScript used for accessing HTML frames.
5. The …….. property of frame object denotes the name of the frame.
6. ……method of frame object removes focus from the object.
7. The ……..object of JavaScript returns useful information about the visitor's browser and system.
8. ……are actions that can be detected by JavaScript.
9. When a confirm box pops up, the user will have to click either "OK" or ……. to proceed.
10. A ……is a small file that the server embeds on the user's computer.

5.9 Summary

JavaScript was designed to add interactivity to HTML pages. It is a scripting language.
A scripting language is a lightweight programming language. A JavaScript consists of lines of executable computer code.

Each object has certain *properties,* and *methods* associated with it. Properties are things that describe the object, and they include sub – objects. For example, is a person is considered to be an object, then hair color, height, etc. are its properties or sub – objects.

There are several objects supported by JavaScript. Some of the important are:

- Window Object
- Frame Object
- Navigator Object
- Document Object

Events are actions that can be detected by JavaScript. Every element on a web page has certain events which can trigger JavaScript functions.

JavaScript supported 3 types of popup boxes. These are:

1. Alert Box
2. Confirm Box
3. Prompt Box

"A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too."

5.10 Self-Assessment Test

Q.1. Briefly describe about JavaScript and write down a program in JavaScript.

Q.2 Discuss about Window object in JavaScript.

Q.3. Compare Frame Object with Navigator Object.

Q.4 Write down a program for creating Prompt Dialog Box.

Q.5 Describe form handling with help of a program.

Q.6 Discuss about Cookies. How will you create the cookies?

5.11 Answers to check your progress

1. JavaScript
2. Object
3. Window
4. Frame
5. name
6. blur()
7. Navigator
8. Events
9. Cancel
10. cookie

5.12 References / Suggested Readings

1. Thomas A Powell, HTML-The Complete Reference,Tata McGraw Hill.
2. Scott Guelich, Shishir Gundavaram, Gunther Birzniek; CGI Programming with Perl 2/e. O'Reilly.
3. Doug Tidwell, James Snell, Pavel Kulchenko; Programming Web Services with SOAP, O'Reilly.
4. Pardi, XML in Action, Web Technology, PHI.
5. Yong, XML Step by Step, PHI.
6. Aaron Weiss, Rebecca Taply, Kim Daniels, Stuven Mulder, Jeff Kaneshki, Web Authoring Desk Reference, Techmedia Publications.
7. HTML The complete Reference, TMH

SUBJECT: WEB DESIGNING	
COURSE CODE: MCA-12	**AUTHOR: DR. SUDHANSHU GAUR**
LESSON NO. 6	
Server-Side Programming	

STRUCTURE

6.0 LEARNING OBJECTIVE

After going through this unit, you will be able to:

• learn the concept of Server-Side Programming.
• learn about Common Gateway Interface.
• learn about ASP and JSP.
• learn about PERL
• learn about VBScript and JavaScript.
• learn about how to configure Web Server.

6.1 INTRODUCTION

This chapter cover the concept of Server-Side Programming. We will go through the concept of Common Gateway Interface. We also have a look at ASP and JSP and with their working. We will discuss about JSP Syntax.

We will have a look at PERL and various variables used in PERL. We will explain VBScript, JavaScript and their use in Server-Side Programming. We also learn about configuring web server with CGI.

6.2 Server-Side Programming

“Server-side scripting is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic html pages.”

It is usually used to provide interactive web sites that interface to databases or other data stores. This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

When the server serves data in a commonly used manner, for example according to the HTTP or FTP protocols, users may have their choice of a number of client programs. In the case of more specialized applications, programmers may write their own server, client, and communications protocol, that can only be used with one another.

Programs that run on a user's local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be considered client-side operations.

Some of the technologies designed mainly or exclusively for server-side scripting, typically by embedding instructions directly in template web pages are as follows:

- CGI (common Gateway Interface)
- ASP (Active Server Pages)
- JSP (Java Server Pages)

6.3 Common Gateway Interface

“The common gateway interface, or cgi, is a standard for external gateway programs to interface with information servers such as http servers.”

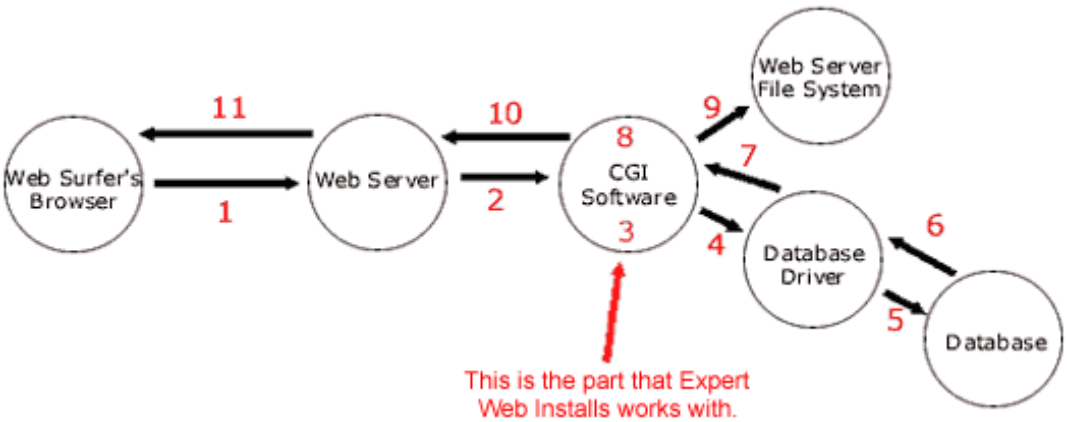
The *Common Gateway Interface (CGI)* is a set of rules that specifies how parameters are passed from programs to Web servers. When a user submits a form, a program may be executed by the Web server, and the results are returned to the browser. The particular

program that is to be executed is specified in the ACTION attribute of the form tag. In general, any program run by a Web server in response to a user's request is called a *script* or *CGI script.*

Common **G**ateway **I**nterface is a specification for transferring information between a World Wide Web server and a CGI program. A CGI program is any program designed to accept and return data that conforms to the CGI specification. The program could be written in any programming language, including C, Perl, Java, or Visual Basic.
CGI programs are the most common way for Web servers to interact dynamically with users. Many HTML pages that contain forms, for example, use a CGI program to process the form's data once it's submitted. Web servers often have a **cgi-bin** directory at the base of the directory tree to hold executable files called with CGI.

Another increasingly common way to provide dynamic feedback for Web users is to include scripts or programs that run on the user's machine rather than the Web server. These programs can be Java applets, Java scripts, or ActiveX controls. These technologies are known collectively as client-side solutions, while the use of CGI is a server-side solution because the processing occurs on the Web server.

6.3.1 CGI processing steps



Each Step mean:

1. The web surfer fills out a form, and clicks submit. The information in the form is sent over the internet to the web server.

2. The web server "grabs" the information from the form, and passes it to the CGI Software.

3. The CGI Software then performs whatever validation of this information that is required. For instance, it checks to see if an email address is valid. If this is a database program, the CGI Software prepares a database statement, to either add, edit or delete information from the database.

4. The CGI Software then executes the prepared database statement, which is passed to the database driver.

5. The database driver acts as a middleman, and performs the requested action on the database itself.

6. The results of the database action are then passed back to the database driver.

7. The database driver sends the information from the database to the CGI Software.

8. The CGI Software takes the information from the database, and manipulates it into the format that is desired.

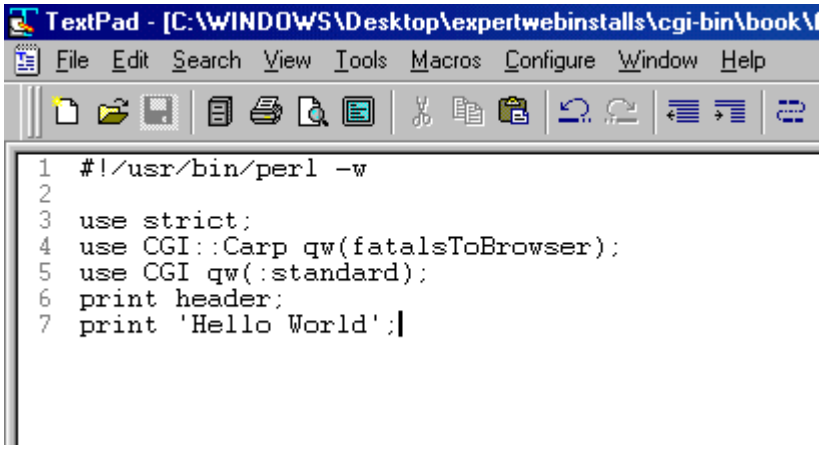
9. If any static html pages need to be created (or similar task needs to be performed), the CGI program accesses the web server computer's file system, and reads, writes, and/or edits files.

10. The CGI Software then sends the result it wants the web surfer's browser to see back to the web server.

11. The web server sends the result it got from the CGI Software back to the web surfer's browser.

6.3.2 Writing first script

- Create a new text file in a plain text editor such as Microsoft Wordpad.
- Name it first.cgi.
- Save this file in plain text only. Do not save it with any special formatting or it won’t work properly.



Explanation of what's going on above:

1. This is the path to the perl interpreter on your system. It varies depending on your web server computer's configuration. The most common location on windows systems is #!c:/perl/bin/perl.exe. Depending on your system configuration, you may not need to put this line at the top of your CGI scripts on windows machines. The –w flag enables warnings, which you should always use so that your code is free of bugs.

2. empty line should always follow the call to the perl interpreter.

3. this requires your code to pre-define variables. Using it helps reduce bugs as well. after every line where you give perl a command such as print, you need to put a semicolon to end the command.

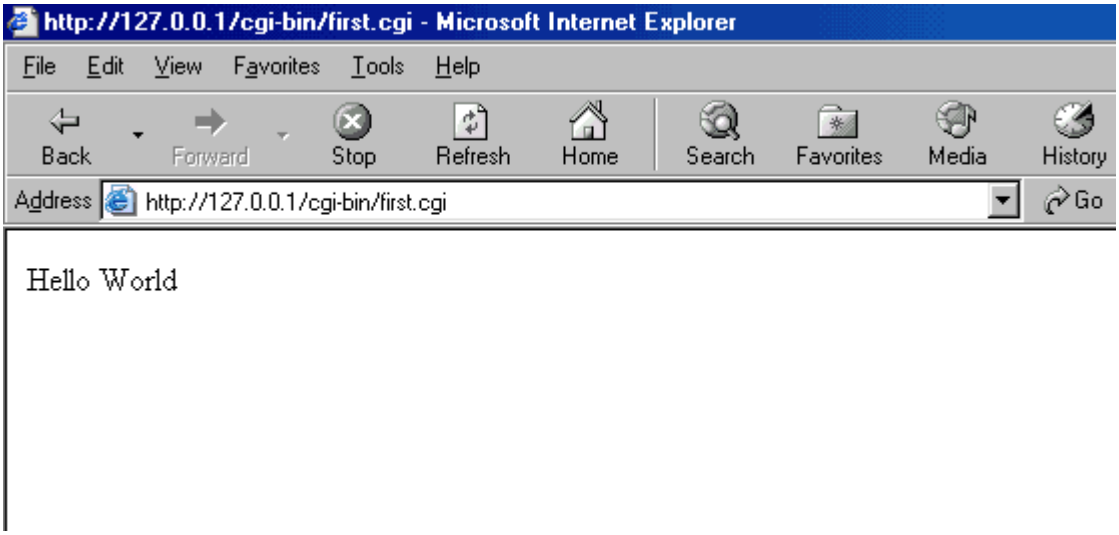
4. this line should always be in a CGI script, as it greatly increases debugging. It will give you a descriptive error message instead of a "500 Internal Server Error".

5. this line loads the CGI.pm module which includes pre-defined functions which are essential to cgi programming.

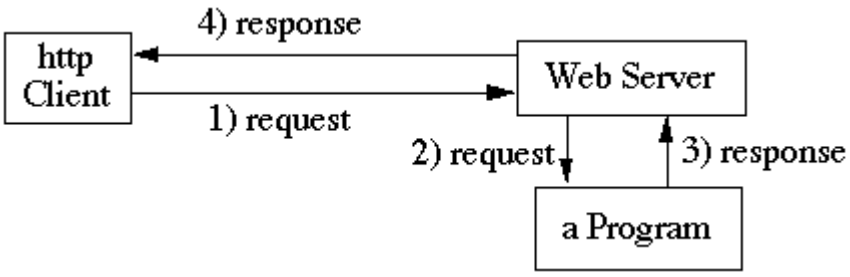
6. This is a function which prints the HTTP Content-type header, which is required in all cgi scripts. It's good to put this line near the top of all your cgi scripts, so you don't forget to include it.

7. This line uses perl's built-in print function to output a string of text to <STDOUT>. <STDOUT> in the case of a cgi script just means 'print it to the browser'. You can use either single or double quotes for this, which does make a difference.

Output:



The Basic Idea



6.4 Active Server Pages

"Active server pages or asp is a technology that enables the designers to make dynamic and interactive web pages**."**

ASP uses server side scripting to dynamically produce web pages that are not affected by the type of browser the web site visitor is using.

The default scripting language used for writing ASP is VBScript, although the designer can use other scripting languages like Jscript (Microsoft's version of JavaScript).

ASP pages have the extension .asp instead of .htm, when a page with the extension .asp is requested by a browser the web server knows to interpret any ASP contained within the

web page before sending the HTML produced to the browser. This way all the ASP is run on the web server and no ASP will ever be passed to the web browser.

Any web pages containing ASP cannot be run by just simply opening the page in a web browser. The page must be requested through a web server that supports ASP, this is why ASP stands for Active Server Pages, no server, no active pages.

What does ASP look like?

The *appearance of an Active Server Page* depends on who or what is viewing it. To the web browser that receives it, an ASP looks just like a normal HTML page. In addition to text and HTML tags, there also exists server side scripts.

The code below shows what a real, live ASP page looks like:

```
<html>
<head>
<title> New Page </title>
</head>
<body>
<h1> My Welcome Page</h1>
<p> <% if Time>#12:00:00 AM# AND_ Time<#12:00:00 PM# Then %> </p>
<h2> Good Morning !!! </h2>
<p> <% if Time>#12:00:00 AM# AND_ Time<#6:00:00 PM# Then %> </p>
<h2> Good Afternoon !!! </h2>
</body></html>
```

6.4.1 Working of ASP

- ASP files are only processed when a client requests them from the server.
- There are some .dll files such as ASP.dll which contains standard objects.
- Functions are called from these dlls.
- Since these functions are previously defined and known, things can be done with ASP are bounded.
- ASP codes do not need to be compiled.

- There exists a global.asa file which is used to specify events and global session variables which are used for all ASP files in the server.
- Before any ASP file is processed, global.asa is processed and it defines global events.
- The processing of an ASP file starts with the separation of ASP scripts and HTML codes in IIS.
- Then each script code is processed.
- The HTML corresponds of these scripts are created, and injected related places.
- Final HTML codes are sent to the client and displayed by the web browser.

What can ASP do?

- The most important feature of ASP is creating dynamic pages.
- You can manage data with database operations such as getting and storing people's information in the Database.
- Interactive applications can be implemented, i.e. news type people like may be displayed.
- Script codes are not sent to the client, which brings security.

Advantages of ASP

- It gives dynamism to the pages created.
- Interactive pages reflecting the preferences of the user can be developed.
- ASP is a very good innovation to dynamic web programming.
- Data obtaining and displaying became more easy and efficient.

Disadvantages of ASP

- It is slower than most of similar technologies.
- Not very much portable.

6.5 Java Server Pages

"Javaserver pages (jsp) is a java technology that allows software developers to dynamically generate html, xml or other types of documents in response to a web client request."

The technology allows java code and certain pre-defined actions to be embedded into static content.."

Features of JSP

JSP gives an ideal platform for creating web applications easily and quickly. The various features supported by JSP are:

1. Platform and Server Independence. The JSP technology follows the 'write once' run anywhere, rule which is the basis of the java language. JSP technology can run on various web server including Apache, Netscape and IIS and is supported by a large variety of tools from different vendors.
2. Environment. JSP uses pure java and takes full advantage of its object-oriented nature. This technology lets the designer separate content generation from layout by accessing component from the page.
3. Extensible JSP Tags. JSP uses a combination of tags and scripting to create dynamic web pages. It allows the designer to extend the JSP tags available. JSP developers can create custom tag libraries, so that more functionality can be extracted using XML-like tags and this leads to less use of scripting in JSP pages.
4. Reusability Across Platform. The JSP pages uses components which are reusable. These reusable components help keep the pages simple and run faster.
5. Easier Maintenance. Application made using JSP technology are easier to maintain.

6.5.1 JSP Processing

6.5.1.1 JSP Architecture

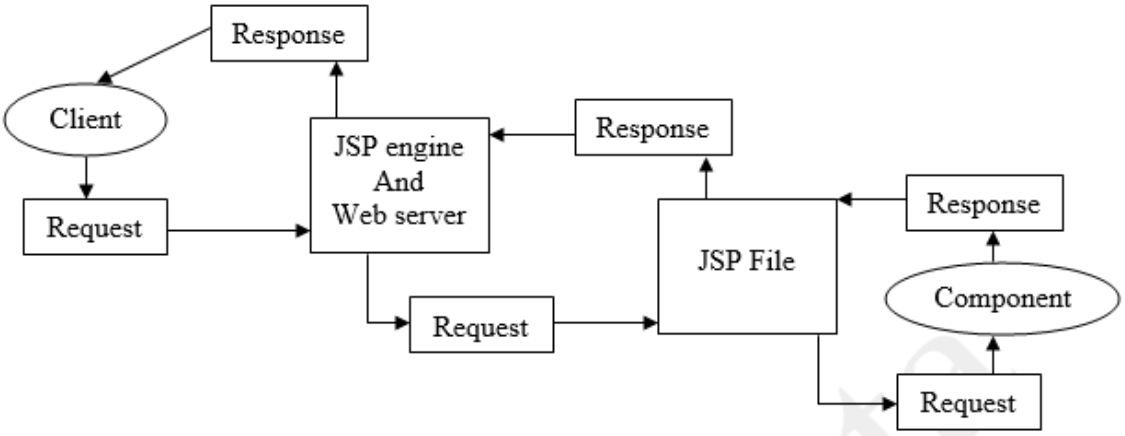
JSP is a part of the Java platform, Enterprise Edition (J2EE), which is the java architecture for developing multitier enterprise applications. A JSP page is executed by a JSP engine, which is installed in a web server or a JSP enabled application server. The JSP engine receives the request from a client to the JSP page and generates responses from the JSP page to the client.

6.5.1.2 Handling JSP Page

JSP page may be a combination of different protocols, components and formats. A JSP page is also a place to enter user information. A user may be asked to enter his home, address, a word or a phrase.

The information the user enters in the form is stored in the request object, which is sent to the JSP engine.

The JSP engine sends the request object to the component (Servlet) the JSP specifies. The component handles the request. The request may be to retrieve certain data or other data store. The component sends the response back to the JSP engine.



The JSP engine passes the response back to the JSP page, where the data is formatted according to the HTML design. The JSP engine and web server then sends the revised JSP page back to the client, where the user can view the results in the web browser. The communication protocol used between the client and server can be HTTP, or it can be some other protocol.

6.5.2 JSP Syntax

Everything in JSP can be broken into two categories:

- Elements: That are processed on the server.
- Template: Data are everything other than elements that the JSP engine ignores.

Element data can be classified into the following categories:

- Directives
- Declarations
- Scriptlets
- Expressions
- Standard Actions

A JSP page contains scripting language and some special JSP tags that can encapsulate tasks that are difficult or time consuming to program.

6.5.2.1 JSP Directives

Directives are instructions for JSP engine that are processed when the JSP page is translated into a servlet. They are used to set global values such as class declarations, methods to be implemented, output content type etc. The directives should start with <%@ and end with %>. There are three types of JSP directives.

They are:

1. The page directive defines a number of attributes that affect the whole page. The syntax is as follows:

<%@ page attributes %>

2. The include directive is used to insert text and code at JSP translation time. The syntax is as follows:

<%@ include file = "relative URL" %>

The file that the file attribute refers to can reference a normal HTML file or another JSP file which will be evaluated at translation time.

3. The taglib directive declares that the page uses custom tags. Uniquely names the tag library defining them and associates a tag prefix that will distinguish the usage of these tags. The syntax is as follows:

<%@ taglib uri = "taglibrary URL prefix" %>

6.5.2.2 JSP Scriptlets

JSP scripting is a mechanism for embedding code fragments directly into an HTML. There are three types of scripting elements, namely scriptlets, expressions and declarations.

Scriptlets are used to embed any piece of java code into the page. The code is inserted into the generated servlet and is executed when the page is requested. The syntax for a scriptlets is as follows:

<% Scriptlet source %>

Scriptlet are executed at request time, when the JSP client processes the client request. If the scriptlet produces output, the output is stored in the JSP writer implicit object *out.*

Writing a simple JSP page using Scriptlets:

Hello.jsp

```
<html>
<head><title> Hello </title></head>
<body>
<%
        int x = 0;
```

```
        for(x = 0; x < 5; x++)
        {
            out.println("<h1> Hello world ! </h1>");
        }
%>
</body>
</html>
```

6.5.2.3 JSP Lifecycle

JSP life cycle can be grouped into seven phases.

1. JSP Page Translation: A java servlet file is generated from the JSP source file. Generated servlet implements the interface javax.servlet.jsp.HttpJspPage. The interface HttpJspPage extends the interface JspPage. This interface JspPage extends the interface javax.servlet.Servlet.
2. JSP Page Compilation: The generated java servlet file is compiled into a java servlet class. The generated servlet class thus implements all the methods of the above said three (javax.servlet.jsp.HttpJspPage, JspPage, javax.servlet.Servlet) interfaces.
3. Class Loading: The java servlet class that was compiled from the JSP source is loaded into the container.
4. Instance Creation: An instance is created for the loaded servlet class. The interface JspPage contains jspInit() and jspDestroy(). The JSP specification has provided a special interface HttpJspPage for JSP pages serving HTTP requests and this interface contains _jspService().
5. jspInit( ) execution: This method is called when the instance is created and it is called only once during JSP life cycle. It is called for the servlet instance initialization.
6. _jspService() execution: This method is called for every request of this JSP during its life cycle. It passes the request and the response objects. _jspService() cannot be overridden.
7. jspDestroy() execution: This method is called when this JSP is destroyed and will not be available for future requests.

6.5.2.4 JSP Expressions (Conditional Processing)

Expressions are used to dynamically calculate values to be inserted directly into the JSP page. These are elements that are evaluated with the result being converted to java.lang.String. After the string is converted, it is written to the out object. The expression should be enclosed within <% = and %>.

While writing an expression, following points should be taken care of:

1. Semicolon cannot be used to end an expression.
2. The expression element can contain any expression that is valid according to java language specification.

An expression can be complex and composed of more than one part or expression. For example, the following shows the date and time that the page requested:

Current Time : <% = new java . util . Date() %>

6.5.2.5 JSP Declarations (Declaring Variables and Methods)

JSP declarations are used to define methods or variables that are to be used in java code later in the JSP file. The declarations get inserted into the main body of the servlet class. It has the following form:

<% ! declarations %>

Since declarations do not generate any output they are usually used with JSP scriptlets or expressions. For example,

<% ! int i = 0; %>

<% ! circle a = new circle(2,0); %>

A declaration element may contain any number of variables or methods.

6.6 PERL (Practical Extraction Report Language)

"Perl is an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information."

It's also a good language for many system management tasks. In computer programming, PERL is high-level, general-purpose, interpreted, dynamic programming language.

Expression syntax corresponds quite closely to C expression syntax. Unlike most Unix utilities, PERL does not arbitrarily limit the size of your data -- if you've got the memory, PERL can slurp in your whole file as a single string. Recursion is of unlimited depth. And the hash tables used by associative arrays grow as necessary to prevent degraded performance. PERL uses sophisticated pattern matching techniques to scan large amounts of data very quickly. Although optimized for scanning text, PERL can also deal with binary data, and can make dbm files look like associative arrays (where dbm is available). Setuid PERL scripts are safer than C programs through a dataflow tracing mechanism which prevents many stupid security holes.

PERL is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

Features

1. The overall structure of PERL derives broadly from C. It is procedural in nature, with variables, expressions, assignment statements, brace-delimited code blocks, control structures, and subroutines.
2. In PERL 5, features were added that support complex data structures, first-class functions (i.e., closures as values), and an object-oriented programming model. These include references, packages, class-based method dispatch, and lexically scoped variables, along with compiler directives. A major additional feature introduced with PERL 5 was the ability to package code as reusable modules.

All versions of PERL do automatic data typing and memory management. The interpreter knows the type and storage requirements of every data object in the program; it allocates and frees storage for them as necessary using reference counting (so it cannot de-allocate circular data structures without manual intervention). Legal type conversions—for example, conversions from number to string—are done automatically at run time; illegal type conversions are fatal errors.

Applications

PERL has many and varied applications, compounded by the availability of many standard and third-party modules.

1. PERL has been used since the early days of the Web to write CGI scripts.
2. PERL is often used as a glue language, tying together systems and interfaces that were not specifically designed to interoperate, and for converting or processing large amounts of data for tasks like creating reports.
3. Graphical user interfaces (GUI's) may be developed using PERL.
4. PERL is also widely used in finance and bioinformatics, where it is valued for rapid application development and deployment, and the ability to handle large data sets.

A perl program is written in file containing commands or statements. Perl is fairly straight forward, widely known and well-respected scripting language. It can be used for a variety of tasks. Some important key points are:

1. perl is case-sensitive language.
2. each perl statement must end with a semi-colon(;).
3. comments should begin with a hash mark (#).
4. braces { } are to group perl statements into a block.
5. perl is free form language and so no constraints on placement of any keywords.

A perl program consists of an ordinary text file containing a series of perl statements with .pl extension. Broad syntactic rules governing where a statement starts and ends are:

1. leading spaces on a line are ignored. You can start a perl statement any where you want: preferred at the beginning of the line.
2. statements are terminated with a semi-colon;
3. spaces, tabs and blank lines outside of string are irrelevant.
4. any thing after a hash sign (#) is ignored except in string.

Example of a perl program:

Hello.pl

```
Print (“hello world \n”);
```

6.6.1 Basic Perl Syntax - Variables: scalars, arrays, hashes.

A comment is any line that begins with a hash sign. It is ignored by the perl interpreter and is used to describe certain parts of code.

For instance:

this is a comment

$this_is_a_variable_definition = ‘some value’; # here is another comment

There are three primary data structures in perl:

1. Scalar
2. Indexed arrays
3. Associative arrays or “hash”.

The scalar:

The scalar can hold one specific piece of information at a time, such as a string or integer. The string has to be enclosed in double quotes (“ “) or single quote (‘ ’). If there is no quote then perl has to decide if the value is a string or integer depending on use.

Strings:

Strings are any sequence of characters that you want to store and/or manipulate.

$string = ‘some value’;

All variables, whether they be a scalar, array, hash, or other type, are CASE-SENSITIVE, meaning that $myvariable and $MYVARIABLE are treated as two different variables.

Scope is a particular area inside opening and closing brackets, and defines where a variable is accessible from.

Consider the following code:

```
$string = ‘David’;
{
  $string = ‘John’;
}
print $string;
```

If you use a single quote, anything you put in the scalar definition goes in just like you typed it. However, if you use a double quote, variables are “interpolated” inside the variable definition.

For instance:

```
$text = ‘cool guy’;
$string = ‘some $text’;
print $string;
# prints literally “some $text”;
```

$text = ‘cool guy’;

$string = “some $text”;

print $string;

prints literally “some cool guy”;

Integers:

Integers in perl are simply numbers, and are the second type of scalar variable. You can use quotes though if you want, perl will figure out what it is depending on what you’re doing with the variable.

$integer = 1;

print $integer; # prints “1”

An easy way to increment (add 1 to) a scalar variable that is an integer is to use the ++ operator.

For instance:

$integer = 1;

$integer++;

print $integer; # prints “2”

Arrays:

Indexed Arrays or Arrays are simply a list of scalar variables. Arrays begin with an @ sign.

@array = (“string 1”, “string 2”, “string 3”);

The opening and closing parentheses are required here.

@array = (“string 1”, “string 2”, “string 3”);

Print $array[0] . “\n”;

Print $array[1] . “\n”;

Print $array[2] . “\n”;

When you’re referencing an individual element of an array, you use a dollar sign, not the @ sign.

Hashes or Associate Arrays:

A hash is a list of name value pairs.

To define a hash, use this syntax (the percent sign):

%hash;

To add new values to a hash, use this syntax:

```
%hash;
$hash{‘key1’} = ‘value1’;
$hash{‘key2} = ‘value2’;
$hash{‘key3’} = ‘value3’;
```

The keys and values are simply scalar variables.

To access hash elements, simply reference the hash key name. Here’s an example:

```
%hash;
$hash{‘key1’} = ‘value1’;
print $hash{‘key1’};
# prints “value1”
```

Printing HTML

Printing HTML to the browser with Perl required escaping quote marks and using the \n character to break a line if you wanted your code to be more readable. You can avoid some of these headaches by using a special print command in Perl. This command allows you to print your HTML as it is written (some special characters still need to be escaped).

Notice the code below that would print out a page with a simple link:

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";
print "<html><head>\n";
print "<title>CGI Test</title>\n";
print "</head>\n";
print "<body><a href=\"http://someplace.com\">Click Here</a>\n";
```

With the need to escape the quotes and add \n characters for source readability, it is a bit tedious to constantly escape quote marks and write in new line characters. An easier way to do this is to use a special print command in Perl:

```
print <<ENDHTML;
....your HTML code here.....

ENDHTML
```

This print command tells the program to print the HTML code until it finds the word ENDHTML again on its own. You can use any word or group of letters you like, just be sure you use the same thing when you want to end the HTML code. For instance, some people use EOM to do this:

```
print <<EOM;
....your HTML code here.....

EOM
```

Just remember to use the exact same thing after the << characters and after the HTML code, and it is case sensitive. Also, when you end the HTML code, you just need your word-- no semicolon. The semicolon only goes with the first print statement.

This allows you to write the code without the need to escape the quote marks, and new lines are created where you break the lines in your code. Here is an example of the last script written in this manner:

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";
print <<ENDHTML;
<html>
<head>
<title>CGI Test</title>
</head>
<body>
<a href="http://someplace.com">Click Here</a>
</body>
</html>

ENDHTML
```

6.7 VBScript

"Vbscript (short for visual basic scripting edition) is an active scripting language developed by microsoft."

When employed in Microsoft Internet Explorer, VBScript is similar in function to JavaScript, as a language to write functions that are embedded in or included from HTML pages and interact with the Document Object Model (DOM) of the page, to perform tasks not possible in HTML alone. Other web browsers such as Firefox, and Opera do not have built-in support for VBScript. This means that where client-side script is required on a web site, developers almost always use JavaScript for cross-browser compatibility.
Besides client-side web development, VBScript is used for server-side processing of web pages, most notably with Microsoft Active Server Pages (ASP).
VBScript can also be used to create applications that run directly on a person's computer running Microsoft Windows.

VBScript provides functions and sub-routines, basic date/time, string manipulation, math, user interaction, error handling, and regular expressions.
Additional functionality can be added through the use of ActiveX technologies.
File system management, file modification, and streaming text operations can be achieved with the Scripting Runtime Library scrrun.dll.

Binary file and memory I/O is provided by the "ADODB.Stream" class, which can also be used as a string builder (since a high amount of VBScript string concatenation is costly due to constant memory re-allocation), and can be used to convert an array of bytes to a string and vice versa.

Database access is made possible through ActiveX Data Objects (ADO), and the IIS Metabase can be manipulated using the GetObject() function with sufficient permissions.

Scripting languages like VBScript and JavaScript, are designed as an extension to HTML. The Web browser receives scripts along with the rest of the web document. It is the browser's responsibility to parse and process the scripts. HTML was extended to include a tag that is used to incorporate scripts into HTML – the <SCRIPT> tag.

For example,

<HTML>
<HEAD>

```
<TITLE>Working with VBScript</TITLE>
<SCRIPT LANGUAGE=”VBScript”>
<!--
        MsgBox “Welcome to my Web page !”
//->
</SCRIPT>
</HEAD>
</HTML>
```

Using VBScript with Forms

Before sending any data from client’s browser to the web server via forms, it is needed to validate data at client’s side itself. VBScript can be used for the purpose very well. Once the form is validated, the same script can be used to forward the data on to the server.

Validating The Form

The Process of validating forms involves checking the form to see if:

- All of the required data is proved
- The data provided is valid.

Checking The Form Input

Checking the Form input may involve determining whether the input given by the client is correct or not. For example, if a web page is about to ask for a password from the user, then if the user enter the correct password, the web page may continue its process. Otherwise, if the user enters a wrong password, the web page may display a message to the user like “The password is Incorrect!!”.

Submitting The Form

Submitting the form after validation involves the use a button that is tied to an event procedure that both validates and at the same time submits the form.

6.8 JavaScript

"Javascript is a scripting language designed to add interactivity to html pages."

A JavaScript consists of lines of executable computer code.

It is usually embedded directly into HTML pages.

It is an interpreted language (means that scripts execute without preliminary compilation).

Everyone can use JavaScript without purchasing a license.

What a JavaScript Can do?

- It gives HTML designers a programming tool.
- It can put dynamic text into an HTML page.
- It can react to events.
- It can read and write HTML elements.
- It can be used to validate data.
- It can be used to detect the visitor's browser.
- It can be used to create cookies.

How to put?

```
<HTML>.
<BODY>
<SCRIPT TYPE="text/javascript">
Document.write("Hello World !")
</SCRIPT>
</BODY>
</HTML>
```

Objects, Properties and Methods in JavaScript

Each object has certain *properties,* and *methods* associated with it. Properties are things that describe the object, and they include sub – objects. For example, is a person is considered to be an object, then hair color, height, etc. are its properties or sub – objects.

Methods are things that the object can do or things that can be done to the object. One method associated with the document object is write(). For example, the document.write() method writes HTML to a Web page. The argument to be passed to write() method is a string of text that will be written. For example, document.write("Hello World!").

6.9 Configuring Web server to support CGI

6.9.1 Configuring Apache Web server to support CGI

By default, the APACHE server's configuration has CGI turned off. Turning it on is a rather simple matter.

Calling scripts with a URL such as:

http://bignosebird.com/cgi-bin/scriptname.cgi

or from within your documents as /cgi-bin/scriptname.cgi

Editing Master Server Configuration file:

Edit the file srm.conf file and do the following:

Find this line.

#ScriptAlias /cgi-bin/ @@ServerRoot@@/cgi-bin/

Change it to read as follows, but of course substitute the full unix path to the directory containing your scripts if it is different than what is shown below.

ScriptAlias /cgi-bin/ /usr/local/apache/share/cgi-bin

Then, track down this line:

#AddHandler cgi-script .cgi

Next, edit the file access.conf file and find this section of code:

<Directory @@ServerRoot@@/cgi-bin>

AllowOverride None

Options None

</Directory>

Change the Directory to the same value that you used when editing the srm.conf file.

After making these changes, restart the server and your scripts should run- provided you did all thing correctly.

6.9.2 Configuring A Windows Server (IIS) To Run CGI Scripts

Setting up IIS to handle files with *.cgi* extension as Perl scripts requires adding a few configuration settings to the server's home directory. Follow these steps:

- 1 Go into the Information Internet Services Administration console.

From the Start menu navigate to the "Control Panel," then "Administrative Tools," and finally "Internet Information Services."

- 2 Open the "Default Web Site" Properties.

Using the navigation on the left-side, navigate through "Local Computer" to "Web Sites" and right-click over "Default Web Site." Choose "Properties."

- 3 Add the Configuration Record to the Home Directory.

Select the Home Directory tab, click Configuration and the Add button. Enter the following values in the form:

  - Executable: #c:\Perl\bin\perl.exe "%s" %s
  - Extension: #.cgi
  - Verbs > Limit to: #GET,HEAD,POST
  - Uncheck the box labeled "Check that file exists."

Press the OK button when you have completed entering the configuration parameters.

6.10 Check Your Progress

1. ……is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic html pages.
2. CGI stands for…….
3. Any program run by a Web server in response to a user's request is called a…….
4. ……is a technology that enables the designers to make dynamic and interactive web pages.
5. Before any ASP file is processed, ……...is processed and it defines global events.
6. ……. Is Data are everything other than elements that the JSP engine ignores.
7. JSP life cycle can be grouped into ……phases.
8. ……is an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information.
9. PERL stands for……...

10. The ……can hold one specific piece of information at a time in PERL.
11. ……are simply a list of scalar variables.

6.11 Summary

“Server-side scripting is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic html pages.”

“the common gateway interface, or cgi, is a standard for external gateway programs to interface with information servers such as http servers.”

“active server pages or asp is a technology that enables the designers to make dynamic and interactive web pages.”

ASP uses server-side scripting to dynamically produce web pages that are not affected by the type of browser the web site visitor is using.
The default scripting language used for writing ASP is VBScript, although the designer can use other scripting languages like Jscript (Microsoft’s version of JavaScript).

The *appearance of an Active Server Page* depends on who or what is viewing it. To the web browser that receives it, an ASP looks just like a normal HTML page. In addition to text and HTML tags, there also exists server side scripts.

“Javaserver pages (jsp) is a java technology that allows software developers to dynamically generate html, xml or other types of documents in response to a web client request.”

Everything in JSP can be broken into two categories:

- Elements: That are processed on the server.
- Template: Data are everything other than elements that the JSP engine ignores.

Element data can be classified into the following categories:

- Directives
- Declarations

- Scriptlets
- Expressions
- Standard Actions

“Perl is an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information.”

“vbscript (short for visual basic scripting edition) is an active scripting language developed by microsoft.”

“javascript is a scripting language designed to add interactivity to html pages.”

A JavaScript consists of lines of executable computer code.
It is usually embedded directly into HTML pages.
It is an interpreted language (means that scripts execute without preliminary compilation).
Everyone can use JavaScript without purchasing a license.

6.12 Self-Assessment Test

Q.1 Explain server-side programming and various languages used for it.
Q.2 Explain CGI working and write down a program for CGI.
Q.3 Describe working of Active Server Pages.
Q.4. Explain JSP Architecture and JSP Lifecycle.
Q.5 Explain various variables used in PERL.
Q.6 Differentiate between JavaScript and VBScript.
Q.7 How will you configure Web Server with CGI ?

6.13 Answers to check your progress

1. Server-side Scripting
2. Common Gateway Interface
3. CGI script
4. Active server pages

5. global.asa
6. Template
7. Seven
8. PERL
9. Practical Extraction Report Language
10. Scalar
11. Arrays

6.14 References / Suggested Readings

1. Thomas A Powell, HTML-The Complete Reference,Tata McGraw Hill.
2. Scott Guelich, Shishir Gundavaram, Gunther Birzniek; CGI Programming with Perl 2/e. O'Reilly.
3. Doug Tidwell, James Snell, Pavel Kulchenko; Programming Web Services with SOAP, O'Reilly.
4. Pardi, XML in Action, Web Technology, PHI.
5. Yong, XML Step by Step, PHI.
6. Aaron Weiss, Rebecca Taply, Kim Daniels, Stuven Mulder, Jeff Kaneshki, Web Authoring Desk Reference, Techmedia Publications.
7. HTML The complete Reference, TMH

SUBJECT: JAVA PROGRAMMING	
COURSE CODE: MCA-12	**AUTHOR: DR. SUDHANSHU GAUR**
LESSON NO. 7	
JSP and ASP	

STRUCTURE

7.0 LEARNING OBJECTIVE

After going through this unit, you will be able to:

- learn about Active Server Pages.
- learn about Java Server Pages.
- learn about various Objects in JSP.
- describe how to retrieve content of HTML form.
- Learn about Cookies.

7.1 INTRODUCTION

This chapter give a detail look about Active Server Pages. We already learn about ASP and JSP briefly in previous chapters. Now we will have detailed look at ASP and JSP.

In this chapter, we will learn about JSP Objects in detail. Various JSP Objects are Request, Response, Session and Application Objects. We will deeply learn about the various methods used in these objects. Here we also learn how these related to HTML and how can we retrieve content of HTML with the help of ASP and JSP.

Although we have learnt about cookies in previous chapters too but we will also have a look at here also. We will have a look at various methods of cookies.

7.2 ACTIVE SERVER PAGES

“Active server pages or asp is a technology that enables the designers to make dynamic and interactive web pages.”

ASP uses server-side scripting to dynamically produce web pages that are not affected by the type of browser the web site visitor is using.
The default scripting language used for writing ASP is VBScript, although the designer can use other scripting languages like Jscript (Microsoft’s version of JavaScript).

7.2.1 Basic Syntax

An ASP file normally contains HTML tags, just like an HTML file. However, an ASP file can also contain server scripts, surrounded by the delimiters <% and %>.
Server scripts are executed on the server, and can contain any expressions, statements, procedures, or operators valid for the scripting language preferred to be used.

7.2.1.1 Write Output to a Browser

The response.write command is used to write output to a browser. The following example sends the text “Hello World !” to the browser:

```
<html>
<body>
<%
Response.write(“Hello World ! ”)
%>
</body>
</html>
```

There is also a shorthand method for the response.write command. The following example also sends the text “Hello World !” to the browser:

```
<html>
<body>
<%=”Hello World ! ”%>
</body>
</html>
```

7.2.2 Lifetime of Variables

A variable declared outside a procedure can be accessed and changed by any script in the ASP file. A variable declared inside a procedure is created and destroyed every time the procedure is executed.

To declare variables accessible to more than one ASP file, declare them as session variables or application variables.

7.2.2.1 Session Variables

Session variables are used to store information about ONE single user, and are available to all pages in one application. Typically, information stored in session variables are name, id, and preferences.

7.2.2.2 Application Variables

Application variables are also available to all pages in one application. Application variables are used to store information about ALL users in a specific application.

Procedures

The ASP source code can contain procedures and functions:

```
<html>
<head>
<% sub vbproc(num1,num2)
Response.write(num1*num2)
end sub
%>
</head>
<body>
<p> Result: <% call vbproc(3,4) %></p>
</body>
</html>
```

Insert the <%@language=”language”%> line above the <html> tag to write procedures or functions in another scripting language than default:

```
<%@ language=”javascript” %>
<html>
<head>
```

```
<%
function jsproc(num1,num2)
{
Response.write(num1*num2)
}
%>
</head>
<body>
<p> Result: <% jsproc(3,4) %></p>
</body>
</html>
```

7.2.3 User Input

The *request* object may be used to retrieve user information from forms.

For example:

```
<form method=”get” action=”simpleform.asp”>
First Name: <input type=”text” name=”fname” /> <br />
Last Name: <input type=”text” name=”lname” /><br /><br />
<input type=”submit” value=”Submit” />
</form>
```

User input can be retrieved in two ways: With Request.QueryString or Request.form

7.2.3.1 Request.QueryString

This command is used to collect values in a form with method= “get”. Information sent from a form with the GET method is visible to everyone (will be displayed in the browser’s address bar) and has limits on the amount of information to send.

If a user typed “pankaj” and “sharma” in the form example above, the URL sent to the server would look like this:

http://www.abes.ac.in/simpleform.asp?fname=pankaj&lname=sharma

Assume that the ASP file “simpleform.asp” contains the following script:

```
<body>
Welcome
<% response.write(request.querystring(“fname”))
```

Response.write(“ ” & request.querystring(“lname”))

%>

</body>

The browser will display the following in the body of the document:

Welcome pankaj sharma

7.2.3.2 Request.Form

This command is used to collect values in a form with method= “post”. Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

If a user typed “Bill” and “Gates” in the form example above, the URL sent to the server would look like this:

Http://www.abes.ac.in/simpleform.asp

Assume that the ASP file “simpleform.asp” contains the following script:

<body>

Welcome

<%

Response.write(request.form(“fname”))

Response.write(“ ” & request.form(“lname”))

%>

</body>

The browser will display the following in the body of the document:

Welcome Bill Gates

7.2.4 Form Validation

User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and reduces the server load.

Using server validation should be considered if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page. The user will then get the error messages on the same page as the form. This makes it easier to discover the error.

7.3 Java Server Pages

An extensible Web technology that uses template data, custom elements, scripting languages, and server-side Java objects to return dynamic content to a client. Typically the template data is HTML or XML elements and The client is often a Web browser.

Java Servlet

A Java program that extends the functionality of a Web server, generating dynamic content and interacting with Web clients using a request-response paradigm.

Static contents

- Typically static HTML page
- Same display for everyone

Dynamic contents

- Contents is dynamically generated based on conditions
- Conditions could be User identity, Time of the day, User entered values through forms and selections

JSP Page

A text-based document capable of returning both static and dynamic content to a client browser. Static content and dynamic content can be intermixed. Static contents are HTML, XML, Text and Dynamic contents are Java code, Displaying properties of JavaBeans, Invoking business logic defined in Custom tags.

Directives

There are five types of JSP directives and scripting elements. With JSP 1.0, most of your JSP is enclosed within a single tag that begins with <% and ends with %>. With the newer JSP 1.1 specification, there are also XML-compliant versions.

JSP directives are for the JSP engine. They do not directly produce any visible output but instead tell the engine what to do with the rest of the JSP page. They are always enclosed within the <%@ … %> tag. The two primary directives are page and include. The taglib directive will not be discussed but is available for creating custom tags with JSP 1.1.

The page directive is the one you'll find at the top of almost all your JSP pages. Although not required, it lets you specify things like where to find supporting Java classes:

<%@ page import="java.util.Date" %>

where to send the surfer in the event of a runtime Java problem:

<%@ page errorPage="errorPage.jsp" %>

and whether you need to manage information at the session level for the user, possibly across multiple Web pages (more later on sessions with JavaBeans):

<%@ page session="true" %>

The include directive lets you separate your content into more manageable elements, such as those for including a common page header or footer. The page included could be a fixed HTML page or more JSP content:

<%@ include file="filename.jsp" %>

Declarations

JSP declarations let you define page-level variables to save information or define supporting methods that the rest of a JSP page may need. If you find yourself including too much code, it is usually better off in a separate Java class. Declarations are found within the <%! … %> tag. Always end variable declarations with a semicolon, as any content must be valid Java statements: <%! int i=0; %>.

Expressions

With expressions in JSP, the results of evaluating the expression are converted to a string and directly included within the output page. JSP expressions belong within <%= … %> tags and do not include semicolons, unless part of a quoted string:

<%= i %>

<%= "Hello" %>

Code Fragments/Scriptlets

JSP code fragments or scriptlets are embedded within <% … %> tags. This Java code is then run when the request is serviced by the Web server. Around the scriptlets would be raw HTML or XML, where the code fragments let you create conditionally executing code, or just something that uses another piece of code. For example, the following displays the string "Hello" within H1, H2, H3, and H4 tags, combining the use of expressions and scriptlets. Scriptlets are not limited to one line of source code:

<% for (int i=1; i<=4; i++){ %>

<h

Comments

The last of the key JSP elements is for embedding comments. Although you can always include HTML comments in your files, users can view these if they view the page's source. If you don't want users to be able to see your comments, you would embed them within the <%-- … --%> tag:

<%-- comment for server side only --%>

7.3.1 Scripts in JSP

A JSP scriptlet is used to contain any code fragment that is valid for the scripting language used in a page. The syntax for a scriptlet is as follows:

```
<%
scripting-language-statements
%>
```

When the scripting language is set to java, a scriptlet is transformed into a Java programming language statement fragment and is inserted into the service method of the JSP page's servlet. A programming language variable created within a scriptlet is accessible from anywhere within the JSP page.

In the web service version of the hello1 application, greeting.jsp contains a scriptlet to retrieve the request parameter named username and test whether it is empty. If the if statement evaluates to true, the response page is included. Because the if statement opens a block, the HTML markup would be followed by a scriptlet that closes the block.

```
<%
String username = request.getParameter("username");
if ( username != null && username.length() > 0 ) {
%>
<%@include file="response.jsp" %>
<%
}
%>
```

7.4 JSP Objects and Components

JSP expressions

If a programmer wants to insert data into an HTML page, then this is achieved by making use of the JSP expression.

The general syntax of JSP expression is as follows:

<%= expression %>

The expression is enclosed between the tags <%= %>

For example, if the programmer wishes to add 10 and 20 and display the result, then the JSP expression written would be as follows:

<%= 10+20 %>

Implicit Objects

Implicit Objects in JSP are objects that are automatically available in JSP. Implicit Objects are Java objects that the JSP Container provides to a developer to access them in their program using JavaBeans and Servlets. These objects are called implicit objects because they are automatically instantiated.

There are many implicit objects available. Some of them are:

Request

The class or the interface name of the object request is http.httpservletrequest. The object request is of type Javax.servlet.http.httpservletrequest. This denotes the data included with the HTTP Request. The client first makes a request that is then passed to the server. The requested object is used to take the value from client's web browser and pass it to the server. This is performed using HTTP request like headers, cookies and arguments.

Response

This denotes the HTTP Response data. The result or the information from a request is denoted by this object. This is in contrast to the request object. The class or the interface name of the object response is http.HttpServletResponse. The object response is of type Javax.servlet.http.httpservletresponse. Generally, the object response is used with cookies. The response object is also used with HTTP Headers.

Session

This denotes the data associated with a specific session of user. The class or the interface name of the object Session is http.HttpSession. The object Session is of type Javax.servlet.http.httpsession. The previous two objects, request and response, are used to pass information from web browser to server and from server to web browser respectively. The Session Object provides the connection or association between the client and the server. The main use of Session Objects is for maintaining states when there are multiple page requests. This will be explained in further detail in following sections.

Out

This denotes the Output stream in the context of page. The class or the interface name of the Out object is jsp.JspWriter. The Out object is written: Javax.servlet.jsp.JspWriter

PageContext

This is used to access page attributes and also to access all the namespaces associated with a JSP page. The class or the interface name of the object PageContext is jsp.pageContext. The object PageContext is written: Javax.servlet.jsp.pagecontext

Application

This is used to share the data with all application pages. The class or the interface name of the Application object is ServletContext. The Application object is written: Javax.servlet.http.ServletContext

Config

This is used to get information regarding the Servlet configuration, stored in the Config object. The class or the interface name of the Config object is ServletConfig. The object Config is written Javax.servlet.http.ServletConfig

Page

The Page object denotes the JSP page, used for calling any instance of a Page's servlet. The class or the interface name of the Page object is jsp.HttpJspPage. The Page object is written: Java.lang.Object

The most commonly used implicit objects are request, response and session objects.

7.5 JSP Configuring

JSP needs any web server; this can be tomcat by apache, WebLogic by bea, or WebSphere by IBM. All jsp should be deployed inside web server. We will use Tomcat server to run JSP, this Tomcat server can run on any platform like windows or linux.

Installation of Tomcat on windows or Installation of Tomcat on linux.

After successful installation of tomcat and JSP we need IDE integrated development environment. These IDE provide software development facilities, help lots in programming. This IDE can contain source code editor, debugger, compiler, automatic generation code tools, and GUI view mode tools which show output at a run-time.

We suggest using, dreamweaver from adobe, or eclipse with myEclipse plugin, NetBeans from sun. Or sun studio creator from sun. These IDEs help in Visual programming

File and folder structure of tomcat

Tomcat
- Bin
- Conf
- Lib
- Logs
- Tmp
- +Webapps
  - Doc
  - Example
  - File
  - Host-manager
  - ROOT
  - jsp
- +Work
  - Catalina

7.5.1 Troubleshooting

Troubleshooting is a form of problem solving most often applied to repair of failed products or processes. It is a logical, systematic search for the source of a problem so that it can be solved, and so the product or process can be made operational again. Troubleshooting is needed to develop and maintain complex systems where the symptoms of a problem can have many possible causes. Troubleshooting is used in many fields such as engineering, system administration, electronics, automotive repair, and diagnostic medicine. Troubleshooting requires identification of the malfunction(s) or symptoms within a system. Then, experience is commonly used to generate possible causes of the symptoms. Determining which cause is most likely is often a process of elimination - eliminating potential causes of a problem. Finally, troubleshooting requires confirmation that the solution restores the product or process to its working state.

In general, troubleshooting is the identification of, or diagnosis of "trouble" in a [system] caused by a failure of some kind. The problem is initially described as symptoms of malfunction, and troubleshooting is the process of determining the causes of these symptoms.

A system can be described in terms of its expected, desired or intended behavior (usually, for artificial systems, its purpose). Events or inputs to the system are expected to generate specific results or outputs. (For example, selecting the "print" option from various computer applications is intended to result in a hardcopy emerging from some specific device). Any unexpected or undesirable behavior is a symptom. Troubleshooting is the process of isolating the specific cause or causes of the symptom. Frequently the symptom is a failure of the product or process to produce any results. (Nothing was printed, for example).

7.6 JSP Objects

7.6.1 JSP Request Objects

The request object in JSP is used to get the values that the client passes to the web server during an HTTP request. The request object is used to take the value from the client’s web browser and pass it to the server. This is performed using an HTTP request such as: headers,

cookies or arguments. The class or the interface name of the object request is http.httpservletrequest.

The object request is written: Javax.servlet.http.httpservletrequest.

Methods of request Object

There are numerous methods available for request object. Some of them are:

- getCookies()
- getHeader(String name)
- getHeaderNames()
- getAttribute(String name)
- getAttributeNames()
- getMethod()
- getParameter(String name)
- getParameterNames()
- getParameterValues(String name)
- getQueryString()
- getRequestURI()
- getServletPath()
- setAttribute(String,Object)
- removeAttribute(String)
-

getCookies()

The getCookies() method of request object returns all cookies sent with the request information by the client. The cookies are returned as an array of Cookie Objects. We will see in detail about JSP cookies in the coming sections.

General syntax of **getHeader()** of request object is as follows:

request.getHeader("String")

getHeader()request object returned value is a string.

For example:

String onlinemca = request.getHeader("onlinemca");

The above would retrieve the value of the HTTP header whose name is onlinemca in JSP.

getHeader(String name)

The method getHeader(String name) of request object is used to return the value of the requested header. The returned value of header is a string.

eneral syntax of getHeader() of request object is as follows:

request.getHeader("String")

In the above the returned value is a String.

For example:

String online = request.getHeader("onlinemca");

The above would retrieve the value of the HTTP header whose name is onlinemca in JSP.

getHeaderNames()

The method getHeaderNames() of request object returns all the header names in the request. This method is used to find available headers. The value returned is an enumerator of all header names.

General syntax of getHeaderNames() of request object is as follows:

request.getHeaderNames();

In the above the returned value is an enumerator.

For example:

Enumeration onlinemca = request.getHeaderNames();

The above returns all header names under the enumerator onlinemca.

getAttribute(String name)

The method getAttribute() of request object is used to return the value of the attribute. The getAttribute() method returns the objects associated with the attribute. When the attribute is not present, then a null value is returned. If the attribute is present then the return value is the object associated with the attribute.

General syntax of getAttribute() of request object is as follows:

request.getAttribute()

In the above the returned value is an object.

For example:

Object onlinemca = request.getAttribute("test");

The above retrieves the object stored in the request test and returns the object in onlinemca.

getAttributeNames()

The method getAttribute() of request object is used to return the object associated with the particular given attribute. If the user wants to get names of all the attributes associated with the current session, then the request object method getAttributeNames() can be used. The returned value is an enumerator of all attribute names.

General syntax of getAttributeNames() of request object is as follows:

request.getAttributeNames()

For example:

Enumeration onlinemca = request.getAttributeNames();

The above returns all attribute names of the current session under the enumerator: onlinemca.

getMethod()

The getMethod() of request object is used to return the methods GET, POST, or PUT corresponding to the requested HTTP method used.

General syntax of getMethod() of request object is as follows:

request.getMethod()

For example:

```
if (request.getMethod().equals("POST"))
{
.....
.....
}
```

In the above example, the method returned by the request.getMethod is compared with POST Method and if the returned method from request.getMethod() equals POST then the statement in if block executes.

getParameter(String name)

getParameter() method of request object is used to return the value of a requested parameter. The returned value of a parameter is a string. If the requested parameter does not exist, then a null value is returned. If the requested parameter exists, then the value of the requested parameter is returned as a string.

General syntax of getParameter() of request object is as follows:

request.getParameter(String name)

The returned value by the above statement is a string.

For example:

String onlinemca = request.getParameter("test");

The above example returns the value of the parameter test passed to the getParameter() method of the request object in the string onlinemca. If the given parameter test does not exist then a null value is assigned to the string onlinemca.

getParameterNames()

The getParameterNames() method of request object is used to return the names of the parameters given in the current request. The names of parameters returned are enumeration of string objects.

General syntax of getParameterNames() of request object is as follows:

request.getParameterNames()

Value returned from the above statement getParameterNames() method is enumeration of string objects.

For example:

Enumeration exforsys = request.getParameterNames();

The above statement returns the names of the parameters in the current request as an enumeration of string object.

getParameterValues(String name)

The getParameter(String name) method of request object was used to return the value of a requested given parameter. The returned value of the parameter is a string. If there are a number of values of parameter to be returned, then the method getParameterValues(String name) of request object can be used by the programmer. The getParameterValues(String name) method of request object is used to return all the values of a given parameter's request. The returned values of parameter is a array of string objects. If the requested parameter is found, then the values associated with it are returned as array of string object. If the requested given parameter is not found, then null value is returned by the method.

General syntax of getParameterValues of request object is as follows:

request.getParameterValues(String name)

The returned value from the above method getParameterValues() is array of string objects.

For example:

String[] vegetables = request.getParameterValues("vegetable");

The above example returns a value of parameter vegetable passed to the method getParameterValues() of request object and the returned values are array of string of vegetables.

getQueryString()

The getQueryString() method of request object is used to return the query string from the request. From this method, the returned value is a string.

General syntax of getQueryString() of request object is as follows:

request.getQueryString()

Value returned from the above method is a string.

For example:

String onlinemca=request.getQueryString();

out.println("Result is"+exforsys);

The above example returns a string exforsys from the method getQueryString() of request object. The value is returned and the string is printed in second statement using out.println statement.

getRequestURI()

The getRequestURI() method of request object is used for returning the URL of the current JSP page. Value returned is a URL denoting path from the protocol name up to query string.

General syntax of getRequestURI() of request object is as follows:

request.getRequestURI()

The above method returns a URL.

For example:

out.println("URI Requested is " + request.getRequestURI());

Output of the above statement would be:

URI Requested is /Jsp/test.jsp

getServletPath()

The getServletPath() method of request object is used to return the part of request URL that calls the servlet.

General syntax of getServletPath() of request object is as follows:

request.getServletPath()

The above method returns a URL that calls the servlet.

For example:

out.println("Path of Servlet is " + request.getServletPath());

The output of the above statement would be:

Path of Servlet is/test.jsp

setAttribute(String,Object)

The setAttribute method of request object is used to set object to the named attribute. If the attribute does not exist, then it is created and assigned to the object.

General syntax of setAttribute of request object is as follows:

request.setAttribute(String, object)

In the above statement the object is assigned with named string given in parameter.

For example:

request.setAttribute("username", "onlinemca");

The above example assigns the value onlinemca to username.

removeAttribute(String)

The removeAttribute method of request object is used to remove the object bound with specified name from the corresponding session. If there is no object bound with specified name then the method simply remains and performs no function.

General syntax of removeAttribute of request object is as follows:

request.removeAttribute(String);

7.6.2 JSP Response Objects

The response object denotes the HTTP Response data. The result or the information of a request is denoted with this object. The response object handles the output of the client. This contrasts with the request object. The class or the interface name of the response object is http.HttpServletResponse.

-The response object is written: Javax.servlet.http.httpservletresponse.

-The response object is generally used by cookies.

-The response object is also used with HTTP Headers.

Methods of response Object

There are numerous methods available for response object. Some of them are:

- setContentType()
- addCookie(Cookie cookie)
- addHeader(String name, String value)
- containsHeader(String name)
- setHeader(String name, String value)
- sendRedirect(String)
- sendError(int status_code)

setContentType()

setContentType() method of response object is used to set the MIME type and character encoding for the page.

General syntax of setContentType() of response object is as follows:

response.setContentType();

For example:

response.setContentType("text/html");

The above statement is used to set the content type as text/html dynamically.

addCookie(Cookie cookie)

addCookie() method of response object is used to add the specified cookie to the response. The addcookie() method is used to write a cookie to the response. If the user wants to add more than one cookie, then using this method by calling it as many times as the user wants will add cookies.

General syntax of addCookie() of response object is as follows:

response.addCookie(Cookie cookie)

For example:

response.addCookie(Cookie exforsys);

The above statement adds the specified cookie exforsys to the response.

addHeader(String name, String value)

addHeader() method of response object is used to write the header as a pair of name and value to the response. If the header is already present, then value is added to the existing header values.

General syntax of addHeader() of response object is as follows:

response.addHeader(String name, String value)

Here the value of string is given as second parameter and this gets assigned to the header given in first parameter as string name.

For example:

response.addHeader("Author", "onlinemca");

The output of above statement is as below:

Author: onlinemca

containsHeader(String name)

containsHeader() method of response object is used to check whether the response already includes the header given as parameter. If the named response header is set then it returns a true value. If the named response header is not set, the value is returned as false. Thus, the containsHeader method is used to test the presence of a header before setting its value. The return value from this method is a Boolean value of true or false.

General syntax of containsHeader() of response object is as follows:

response.containsHeader(String name)

Return value of the above containsHeader() method is a Boolean value true or false.

setHeader(String name, String value)

setHeader method of response object is used to create an HTTP Header with the name and value given as string. If the header is already present, then the original value is replaced by the current value given as parameter in this method.

General syntax of setHeader of response object is as follows:

response.setHeader(String name, String value)

For example:

response.setHeader("Content_Type","text/html");

The above statement would give output as

Content_Type: text/html

sendRedirect(String)

sendRedirect method of response object is used to send a redirect response to the client temporarily by making use of redirect location URL given in parameter. Thus the sendRedirect method of the response object enables one to forward a request to a new target. But one must note that if the JSP executing has already sent page content to the client, then the sendRedirect() method of response object will not work and will fail.

General syntax of sendRedirect of response object is as follows:

response.sendRedirect(String)

In the above the URL is given as string.

For example:

response.sendRedirect("http://xxx.test.com/error.html");

The above statement would redirect response to the error.html URL mentioned in string in Parameter of the method sendRedirect() of response object.

sendError(int status_code)

sendError method of response object is used to send an error response to the client containing the specified status code given in parameter.

General syntax of sendError of response object is as follows:

response.sendError(int status_code)

7.6.3 JSP Session Object

Session Object denotes the data associated with a specific session of user. The class or the interface name of the object session is http.HttpSession. The object session is written as: Javax.servlet.http.httpsession.

The previous two objects, request and response, are used to pass information from web browser to server and from server to web browser respectively. But the Session Object provides the connection or association between the client and the server. The main use of Session Objects is to maintain states when there are multiple page requests.

The main feature of session object is to navigate between multiple pages in a application where variables are stored for the entire user session. The session objects do not lose the variables and the value remains for the user' session. The concept of maintenance of sessions can be performed by cookies or URL rewriting. A detailed approach of session handling will be discusses in coming sections.

Methods of Session Object

There are numerous methods available for session Object. Some are:

- getAttribute(String name)

- getAttributeNames
- isNew()
- getCreationTime
- getId
- invalidate()
- getLastAccessedTime
- getMaxInactiveInterval
- removeAttribute(String name)
- setAttribute(String, object)

getAttribute(String name)

The getAttribute method of session object is used to return the object with the specified name given in parameter. If there is no object then a null value is returned.

General syntax of getAttribute of session object is as follows:

session.getAttribute(String name)

The value returned is an object of the corresponding name given as string in parameter. The returned value from the getAttribute() method is an object written: java.lang.Object.

For example

String exforsys = (String) session.getAttribute("name");

In the above statement, the value returned by the method getAttribute of session object is the object of name given in parameter of type java.lang. Object and this is typecast to String data type and is assigned to the string exforsys.

getAttributeNames

The getAttributeNames method of session object is used to retrieve all attribute names associated with the current session. The name of each object of the current session is returned. The value returned by this method is an enumeration of objects that contains all the unique names stored in the session object.

General Syntax

session.getAttributeNames()

The returned value by this method getAttributeNames() is Enumeration of object.

For example

exforsys = session.getAttributeNames( )

The above statement returns enumeration of objects, which contains all the unique names stored in the current session object in the enumeration object exforsys.

isNew()

The isNew() method of session object returns a true value if the session is new. If the session is not new, then a false value is returned. The session is marked as new if the server has created the session, but the client has not yet acknowledged the session. If a client has not yet chosen the session, i.e., the client switched off the cookie by choice, then the session is considered new. Then the isNew() method returns true value until the client joins the session. Thus, the isNew() method session object returns a Boolean value of true of false.

General syntax of isNew() of session object is as follows:

session.isNew()

The returned value from the above method isNew() is Boolean.

7.6.4 JSP Application Objects

Application Object is used to share the data with all application pages. Thus, all users share information of a given application using the Application object. The Application object is accessed by any JSP present in the application. The class or the interface name of the object application is ServletContext.

The application object is written as:

Javax.servlet.http.ServletContext

Methods of Application Object

There are numerous methods available for Application object. Some of the methods of Application object are:

- getAttribute(String name)
- getAttributeNames
- setAttribute(String objName, Object object)
- removeAttribute(String objName)
- getMajorVersion()
- getMinorVersion()
- getServerInfo()
- getInitParameter(String name)
- getInitParameterNames

- getResourceAsStream(Path)
- log(Message)

getAttribute(String name)

The method getAttribute of Application object is used to return the attribute with the specified name. It returns the object given in parameter with name. If the object with name given in parameter of this getAttribute does not exist, then null value is returned.

General syntax of getAttribute method of Application object is as follows:

application.getAttribute(String name);

For example:

application.getAttribute("onlinemca");

The above statement returns the object onlinemca.

getAttributeNames

The method getAttributeNames of Application object is used to return the attribute names available within the application. The names of attributes returned are an Enumeration.

General syntax of getAttributeNames method of Application object is as follows:

application.getAttributeNames();

For example:

Enumeration onlinemca;

onlinemca=application.getAttributeNames();

The above example returns the attribute names available within the current application as enumeration in onlinemca.

setAttribute(String objName, Object object)

The method setAttribute of Application object is used to store the object with the given object name in the application.

General syntax of setAttribute method of Application object is as follows:

application.setAttribute(String objName, Object object);

The above syntax stores the objname mentioned in String in the corresponding object mentioned as Object in the parameter of the setAttribute method.

For example:

application.setAttribute("exvar", "onlinemca");

In the above example, the object exvar is stored with the object name onlinemca in the application.

removeAttribute(String objName)

The method removeAttribute of Application object is used to remove the name of the object mentioned in parameter of this method from the object of the application.

General syntax of removeAttribute method of Application object is as follows:

application.removeAttribute(String objName);

For example:

application.setAttribute("password",password);

application.removeAttribute("password");

The above statement removes the name from the object password of the application.

getMajorVersion()

The method getMajorVersion of Application object is used to return the major version of the Servlet API for the JSP Container.

General syntax of getMajorVersion method of Application object is as follows:

application.getMajorVersion();

The returned value from the above method is an integer denoting the major version of the Servlet API.

For example:

out.println("Major Version:"+application.getMajorVersion());

Major Version:2

The above statement gives 2 as the major version of the Servlet API in use for the Application object.

getMinorVersion():

The method getMinorVersion of Application object is used to return the minor version of the Servlet API for the JSP Container.

General syntax of getMinorVersion method of Application object is as follows:

application.getMinorVersion();

The returned value from the above method is an integer denoting the minor version of the Servlet API.

For example:

out.println("Minor Version:"+application.getMinorVersion());

Minor Version:1

The above gives 1 as the minor version of the Servlet API in use for the Application object.

getServerInfo()

The method getServerInfo of Application object is used to return the name and version number of the JRun servlet engine. Information about the JSP Container, such as, the name and product version, are returned by the method getServerInfo of Application object.

General syntax of getServerInfo method of Application object is as follows:

application.getServerInfo();

For example:

out.println("Server Information:"+application.getServerInfo());

getInitParameter(String name)

The method getInitParameter of Application object is used to return the value of an initialization parameter. If the parameter does not exist, then null value is returned.

General syntax of getInitParameter method of Application object is as follows:

application.getInitParameter(String name);

For example:

String onlinemca = application.getInitParameter("eURL");

In the above, the value of initialization parameter eURL is retrieved and stored in string onlinemca.

getInitParameterNames

The method getInitParameterNames of Application object is used to return the name of each initialization parameter. The returned value is an enumeration. General syntax of getInitParameterNames method of Application object is as follows:

application.getInitParameterNames();

The returned value from the above method is an enumeration.

For example:

Enumeration e;

e=application.getInitParameterNames();

getResourceAsStream(Path)

The method getResourceAsStream of Application object is used to translate the resource URL mentioned as parameter in the method into an input stream to read. General syntax of getResourceAsStream method of Application object is as follows:

application.getResourceAsStream(Path);

For example:

InputStream stream = application.getResourceAsStream("/onlinemca.txt");

The above example translates the URL /onlinemca.txt mentioned in the parameter of getResourceAsStream method into an input stream to read.

log(Message)

The method log of Application object is used to write a text string to the JSP Container's default log file.

General syntax of log method of Application object is as follows:

application.log(Message);

7.7 Retrieving the Contents of a HTML form

Forms are, of course, the most important way of getting information from the customer of a web site. In this section, we'll just create a simple color survey and print the results back to the user.

First, create the entry form. Our HTML form will send its answers to form.jsp for processing.

For this example, the name="name" and name="color" are very important. You will use these keys to extract the user's responses.

form.html

```
<form action="form.jsp" method="get">
<table>
<tr><td><b>Name</b>
<td><input type="text" name="name">
<tr><td><b>Favorite color</b>
<td><input type="text" name="color">
</table>
```

<input type="submit" value="Send">
</form>

Keeps the browser request information in the request object. The request object contains the environment variables you may be familiar with from CGI programming. For example, it has the browser type, any HTTP headers, the server name and the browser IP address.
You can get form values using request.getParameter object.
The following JSP script will extract the form values and print them right back to the user.

form.jsp

Name: <%= request.getParameter("name") %>
Color: <%= request.getParameter("color") %>

7.7.1 Retrieving a Query String

An include action executes the included JSP page and appends the generated output onto its own output stream. Request parameters parsed from the URL's query string are available not only to the main JSP page but to all included JSP pages as well. It is possible to temporarily override a request parameter or to temporarily introduce a new request parameter when calling a JSP page. This is done by using the jsp:param action.
In this example, param1 is specified in the query string and is automatically made available to the callee JSP page. param2 is also specified in the query string but is overridden by the caller. Notice that param2 reverts to its original value after the call. param3 is a new request parameter created by the caller. Notice that param3 is only available to the callee and when the callee returns, param3 no longer exists. Here is the caller JSP page:

```
<html>
<head></head>
<body>

<jsp:include page="callee.jsp" />
    <jsp:param name="param2" value="value2" />
    <jsp:param name="param3" value="value3" />
</jsp:include>

Caller:
param1: <%= request.getParameter("param1") %>
param2: <%= request.getParameter("param2") %>
param3: <%= request.getParameter("param3") %>

</body>
</html>
```

Here is the JSP page being called:

```
Callee:
param1: <%= request.getParameter("param1") %>
param2: <%= request.getParameter("param2") %>
param3: <%= request.getParameter("param3") %>
```

If the example is called with the URL:

http://hostname.com?param1=a¶m2=b

the output would be:

```
<html>
<head></head>
<body>

<jsp:include page="callee.jsp" />
    <jsp:param name="param2" value="value2" />
    <jsp:param name="param3" value="value3" />
</jsp:include>

Caller:
param1: <%= request.getParameter("param1") %>
param2: <%= request.getParameter("param2") %>
param3: <%= request.getParameter("param3") %>

</body>
</html>
```

Here is the JSP page being called:

```
Callee:
param1: <%= request.getParameter("param1") %>
param2: <%= request.getParameter("param2") %>
param3: <%= request.getParameter("param3") %>
```

7.8 Cookies

Cookies are short pieces of data sent by web servers to the client browser. The cookies are saved to clients hard disk in the form of small text file. Cookies helps the web servers to identify web users, by this way server tracks the user. Cookies pay very important role in the session tracking.

Cookie Class

In JSP cookie are the object of the class javax.servlet.http.Cookie. This class is used to creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server. A cookie's value can uniquely identify a client, so cookies are commonly used for session management. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

The getCookies() method of the request object returns an array of Cookie objects. Cookies can be constructed using the following code:
Cookie(java.lang.String name, java.lang.String value)

Methods of Cookie objects

getComment()

Returns the comment describing the purpose of this cookie, or null if no such comment has been defined.

getMaxAge()

Returns the maximum specified age of the cookie.

getName()

Returns the name of the cookie.

getPath()

Returns the prefix of all URLs for which this cookie is targeted.

getValue()

Returns the value of the cookie.

setComment(String)

If a web browser presents this cookie to a user, the cookie's purpose will be described using this comment.

setMaxAge(int)

Sets the maximum age of the cookie. The cookie will expire after that many seconds have passed. Negative values indicate the default behavior: the cookie is not stored persistently, and will be deleted when the user web browser exits. A zero value causes the cookie to be deleted

setPath(String)

This cookie should be presented only with requests beginning with this URL.

setValue(String)

Sets the value of the cookie. Values with various special characters (white space, brackets and parentheses, the equals sign, comma, double quote, slashes, question marks, the "at" sign, colon, and semicolon) should be avoided. Empty values may not behave the same way on all browsers.

 # Cookies also covered in Chapter-5

7.9 Check Your Progress

1. The default scripting language used for writing ASP is…….
2. The ……command is used to write output to a browser.
3. ……..variables are used to store information about ONE single user.
4. ……..is a Java program that extends the functionality of a Web server
5. JSP code …….are embedded within <% … %> tags.
6. The class or the interface name of the object request is……..
7. The class or the interface name of the object session is……...

8. ……...are short pieces of data sent by web servers to the client browser
9. In JSP cookie are the object of the class ……...
10. ……function Returns the maximum specified age of the cookie

7.10 Summary

"ACTIVE SERVER PAGES OR ASP IS A TECHNOLOGY THAT ENABLES THE DESIGNERS TO MAKE DYNAMIC AND INTERACTIVE WEB PAGES."

ASP uses server-side scripting to dynamically produce web pages that are not affected by the type of browser the web site visitor is using.
The default scripting language used for writing ASP is VBScript, although the designer can use other scripting languages like Jscript (Microsoft's version of JavaScript).

A variable declared outside a procedure can be accessed and changed by any script in the ASP file. A variable declared inside a procedure is created and destroyed every time the procedure is executed.
To declare variables accessible to more than one ASP file, declare them as session variables or application variables.

An extensible Web technology that uses template data, custom elements, scripting languages, and server-side Java objects to return dynamic content to a client. Typically the template data is HTML or XML elements and The client is often a Web browser.

Implicit Objects in JSP are objects that are automatically available in JSP. Implicit Objects are Java objects that the JSP Container provides to a developer to access them in their program using JavaBeans and Servlets. These objects are called implicit objects because they are automatically instantiated.

JSP needs any web server; this can be tomcat by apache, WebLogic by bea, or WebSphere by IBM. All jsp should be deployed inside web server. We will use Tomcat server to run JSP, this Tomcat server can run on any platform like windows or linux.

Main objects of JSP are : Request Object, Response Object, Session Object and Application Objects.

Forms are, of course, the most important way of getting information from the customer of a web site. In this section, we'll just create a simple color survey and print the results back to the user.

First, create the entry form. Our HTML form will send its answers to form.jsp for processing.

Cookies are short pieces of data sent by web servers to the client browser. The cookies are saved to clients hard disk in the form of small text file. Cookies helps the web servers to identify web users, by this way server tracks the user. Cookies pay very important role in the session tracking.

7.11 Self-Assessment Test

Q.1 Explain ASP and different types of variables in ASP.

Q.2 Describe User-Input in Active Server Pages.

Q.3 Explain JSP in detail.

Q.4 Explain Request Objects and Response Objects.

Q.5 Explain the working of retrieving of HTML content.

Q.6 Differentiate between Session and Application Objects in JSP.

Q.7 Write a short note on cookies.

7.12 Answers to check your progress

1. VBScript
2. response.write
3. Session
4. Java Servlet
5. fragments or scriptlets
6. http.httpservletrequest
7. http.HttpSession
8. Cookies

9. javax.servlet.http.Cookie

10. getMaxAge()

7.13 References / Suggested Readings

1. Thomas A Powell, HTML-The Complete Reference, Tata McGraw Hill.
2. Scott Guelich, Shishir Gundavaram, Gunther Birzniek; CGI Programming with Perl 2/e. O'Reilly.
3. Doug Tidwell, James Snell, Pavel Kulchenko; Programming Web Services with SOAP, O'Reilly.
4. Pardi, XML in Action, Web Technology, PHI.
5. Yong, XML Step by Step, PHI.
6. Aaron Weiss, Rebecca Taply, Kim Daniels, Stuven Mulder, Jeff Kaneshki, Web Authoring Desk Reference, Techmedia Publications.
7. HTML The complete Reference, TMH