Definition Web of Engineering

Based on this definition and on (Deshpande et al. 2002) we define Web Engineering as follows:
1) Web Engineering is the application of systematic and quantifiable approaches (concepts, methods, techniques, tools) to cost-effective requirements analysis, design, implementation, testing, operation, and maintenance of high-quality Web applications.
2) Web Engineering is also the scientific discipline concerned with the study of these approaches.

Actually traditional web applications are developed in random order without any systematic approach. One can say that web applications are developed in an ad hoc manner. There are lots of problem in traditional web application. Now to solve the all traditional problems there is a need of engineered development of web applications. So here is the subject web engineering. you can also say that web site engineering.

Now the web applications are developed from the point of view of Software Engineering. In the definition the term analysis, design, implementation, testing, operation, and maintenance are taken from Software Engineering.

Web Engineering as a discipline:

Proponents of Web engineering supported the establishment of Web engineering as a discipline at an early stage of Web. First Workshop on Web Engineering was held in conjunction with World Wide Web Conference held in Brisbane, Australia, in 1998. San Murugesan, Yogesh Deshpande, Steve Hansen and Athula Ginige, from University of Western Sydney, Australia formally promoted Web engineering as a new discipline in the first ICSE workshop on Web Engineering in 1999. Since then they published a series of papers in a number of journals, conferences and magazines to promote their view and got wide support. Major arguments for Web engineering as a new discipline are:

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

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

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

Modeling disciplines
• Design Manufacturing of Steel Plant equipments
• Process Modelling of Web applications
• Requirements Engineering for Web applications
• B2B applications

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

Implementation disciplines
• Integrated Web Application Development Environments
• Code Generation for Web Applications
• Software Factories for/on the Web
• Web 2.0, AJAX, E4X, ASP.NET, PHP and Other New Developments
• Web Services Development and Deployment

Testing disciplines
• Testing and Evaluation of Web systems and Applications
• Testing Automation, Methods and Tools

Applications categories disciplines
• Semantic Web applications
• Ubiquitous and Mobile Web Applications
• Mobile Web Application Development
• Device Independent Web Delivery
• Localization and Internationalization Of Web Applications

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

Content-related

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

Categories of Web Applications

Below fig identifies different categories of Web applications depending on their development history and their degree of complexity and gives examples (cf. Murugesan 2000), and(Gerti Kappel)

1. Document Centric (Static homepage, web radio, company web site)

2. Interactive (Virtual exhibition, news site, travel planning)

3. Transactional (online banking, shopping, booking system)

4. Workflow based (E government, B2B solution)

5. Collaborative (chat room, E learning plateform, P2P-services)

6. Portal oriented (community portal, online shopping mall, business portal)

7. Ubiquitous (customized services, location aware services, Multi plateform delivery)

8. Semantic (Knowledge management, syndication, recomender system)

9. Social (web logs, collaborative filtering, Virtual shared workplace)
Subject: web engineering

Lecture 1

Topic: Information Architecture: The role of the Information Architect

Information architecture (IA) is the art of expressing a model or concept of information used in activities that require explicit details of complex systems. Among these activities are library systems, Content Management Systems, web development, user interactions, database development, programming, technical writing, enterprise architecture, and critical system software design. Information architecture has somewhat different meanings in these different branches of IS or IT architecture. Most definitions have common qualities: a structural design of shared environments, methods of organizing and labeling websites, intranets, and online communities, and ways of bringing the principles of design and architecture to the digital landscape.

Role of information architect:
An Information Architect organizes a website so that users have a better online experience. In general, their main responsibilities are to:

- Assign tasks to team members. The Information Architect often doubles up as the Project Manager.
- Capture the site’s design goals.
- Communicate the business objectives, such as the site’s sales targets, audience, and language requirements.
- Create access points to content from different in-coming pages.
- Design the navigation system, menus, sitemaps etc.
- Label and organize data.
- Map content to the appropriate section.
- Protect users from getting lost on the site.
- Before any coding begins, the Information Architect meets the client and defines the project’s scope, objectives and target audience.

Documentation of Success Criteria

The meeting minutes are then returned to the client for confirmation. Once confirmed, they’re circulated to all members involved in the development process.

When the project enters the production stage, the Information Architect works with the web designers to develop the interface, icons and ensure the navigation systems are integrated correctly with the overall site architecture.
Subject: web engineering

For very complicated sections, the Information Architect and Software Engineers work together to ensure that each site component make sense so that the user can easily achieve their goal.

The Information Architect communicates with the team during all key stages in the development cycle. On small projects the Information Architect may perform Project Management duties as these two areas frequently overlap. It is imperative to record client feedback at all stages and circulate it accordingly.

Communication

Lack of planning at the kickoff phase often results in untold disasters at later stages - often with serious financial repercussions.

This may occur when, for example, the person delegated to lead the project lacks sufficient technical understanding to extract relevant information from the client. The Information Architect has this knowledge and can ask key questions that others will have overlooked.

Finally, the Information Architect also works with the Quality Control team to ensure that the site is performing correctly and, for example, by analyzing the log files, identify areas where users are struggling to locate date or getting lost.
Subject: web engineering

Lecture 2

Topic: Collaboration and Communication

The information architect must communicate effectively with the web site development team. This is challenging, since an 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.

In the early days of the Web, web sites were often designed, built, and managed by a single individual through sheer force of will. This webmaster was responsible for assembling 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 graphic designers, then editors, then programmers.

Then companies began to demand more of their sites and, consequently, of their webmasters. Simple home pages quickly evolved into complex web sites. People wanted more content, better organization, greater function, and prettier graphics. Extensions, plug-ins, and languages proliferated. Tables, VRML, frames, Shockwave, Java, and ActiveX were added to the toolbox. No mortal webmaster could keep up with the rising expectations and the 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. An individual cannot be an expert in all facets of the process. Rather, a team of individuals with complementary areas of expertise must work together. 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, writing and editing, 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 to the web site and what will bring them back again.

Information Architecture
The information architects focus on the design of organization, indexing, labeling, and navigation systems to support browsing and searching throughout the web site.

**Graphic Design**

The designers are responsible for the graphic design and page layout that defines the graphic identity or look of the web site. They strive to create and implement a design philosophy that balances form and function.

**Editorial**

Editors focus on the use of language throughout the web site. Their tasks may involve proofreading and editing copy, massaging content to ensure a common voice for the site, and creating new copy.

**Technical**

The technical designers and programmers are responsible for server administration and the development or integration of site production tools and web site applications. They advise the other teams regarding technology-related opportunities and limitations.

**Project Management**

The project manager keeps the project on schedule and within budget. He or she facilitates communication between the other teams and the clients or internal stakeholders.

The success of a web site design and production project depends on successful communication and collaboration between these specialized team members. A linear, black-box, throw-it-over-the-wall methodology just won't work. Everyone needs to understand the goals, perspectives, and approaches of the other members of the team. For example, while the marketing specialist may lead the audience analysis process, he or she needs to anticipate the types of questions about the audience that the specialists will have. Otherwise, each will need to start from scratch in learning about that audience, wasting substantial time and resources.
Lecture 3

Topic: Organizing Information, Organizational Challenges

We organize to understand, to explain, and to control. Our classification systems inherently reflect social and political perspectives and objectives. We live in the first world. They live in the third world. She is a freedom fighter. He is a terrorist. The way we organize, label, and relate information influences the way people comprehend that information.

As information architects, we organize information so that people can find the right answers to their questions. We strive to support casual browsing and directed searching. Our aim is to apply organization and labeling systems that make sense to users.

The Web provides us with a wonderfully flexible environment in which to organize. We can apply multiple organization systems to the same content and escape the physical limitations of the print world. So why are many large web sites so difficult to navigate? Why can't the people who design these sites make it easy to find information? These common questions focus attention on the very real challenge of organizing information.

Organizational Challenges

In recent years, increasing attention has been focused on the challenge of organizing information. Yet, this challenge is not new. People have struggled with the difficulties of information organization for centuries. The field of librarianship has been largely devoted to the task of organizing and providing access to information. This quiet yet powerful revolution is driven by the decentralizing force of the global Internet. Not long ago, the responsibility for labeling, organizing, and providing access to information fell squarely in the laps of librarians. They classified, cataloged, and helped us find the information we needed.

The Internet is forcing the responsibility for organizing information on more of us each day. How many corporate web sites exist today? How many personal home pages? What about tomorrow? As the Internet provides us all with the freedom to publish information, it quietly burdens us with the responsibility to organize that information. As we struggle to meet that challenge, we unknowingly adopt the language of librarians. How should we label that content? Is there an existing classification system? Who's going to catalog all of that information?

We're moving towards a world where tremendous numbers of people publish and organize their own information. The challenges inherent in organizing that information
Subject: web engineering

become more recognized and more important. Let's explore some of the reasons why organizing information in useful ways is so difficult

Heterogeneity

Heterogeneity refers to an object or collection of objects composed of unrelated or unlike parts. You might refer to grandma's homemade broth with its assortment of vegetables, meats, and other mysterious leftovers as heterogeneous. At the other end of the scale, homogeneous refers to something composed of similar or identical elements. For example, Oreo cookies are homogeneous. Every cookie looks and tastes the same. Most web sites are highly heterogeneous in two respects. First, web sites often provide access to documents and their components at varying levels of granularity. A web site might present articles and journals and journal databases side by side. Links might lead to pages, sections of pages, or to other web sites. Second, web sites typically provide access to documents in multiple formats. The heterogeneous nature of web sites makes it difficult to impose highly structured organization systems on the content. It doesn't make sense to classify documents at varying levels of granularity side by side. An article and a magazine should be treated differently. Similarly, it may not make sense to handle varying formats the same way. Each format will have uniquely important characteristics.

Differences in Perspectives that labeling and organization systems are intensely affected by their creators' perspectives. We see this at the corporate level with web sites organized according to internal divisions or org charts. In these web sites, we see groupings 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 organization 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 power or respect. Because of the inherent power of information organization in forming understanding and opinion, the process of designing information architectures for web sites and intranets can involve a strong undercurrent of politics. The choice of organization and labeling systems can have a big impact on how users of the site perceive the company, its departments, and its products.
Lecture 4

Topic: Organizing Web sites and Intranets

Organizing Web Sites and Intranets:

The organization of information in web sites and intranets is a major factor in determining success, and yet many web development teams lack the understanding necessary to do the job well. Our goal in this chapter is to provide a foundation for tackling even the most challenging information organization projects.

Organization systems are composed of organization schemes and organization structures. An organization scheme defines the shared characteristics of content items and influences the logical grouping of those items. An organization structure defines the types of relationships between content items and groups.

Before diving in, it's important to understand information organization in the context of web site development. Organization is closely related to navigation, labeling, and indexing. The hierarchical organization structures of web sites often play the part of primary navigation system. The labels of categories play a significant role in defining the contents of those categories. Manual indexing is ultimately a tool for organizing content items into groups at a very detailed level. Despite these closely knit relationships, it is both possible and useful to isolate the design of organization systems, which will form the foundation for navigation and labeling systems. By focusing solely on the logical grouping of information, you avoid the distractions of implementation details and design a better web site.

Organization Schemes

We navigate through organization schemes every day. Phone books, supermarkets, and television programming guides all use organization schemes to facilitate access. Some schemes are easy to use. We rarely have difficulty finding a friend's phone number in the alphabetical organization scheme of the white pages. Some schemes are intensely frustrating. Trying to find marshmallows or popcorn in a large and unfamiliar supermarket can drive us crazy. In fact, the organization schemes of the phone book and the supermarket are fundamentally different. The alphabetical organization scheme of the phone book's white pages is exact. The hybrid topical/task-oriented organization scheme of the supermarket is ambiguous.

Alphabetical:
Subject: web engineering

An alphabetical organization scheme is the primary organization scheme for encyclopedias and dictionaries

Chronological

Certain types of information lend themselves to chronological organization. For example, an archive of press releases might be organized by the date of release

Geographical

Place is often an important characteristic of information. With the exception of border disputes, geographical organization schemes are fairly straightforward to design and use. Organization structure plays an intangible yet very important role in the design of web sites. While we interact with organization structures every day, we rarely think about them. Movies are linear in their physical structure. We experience them frame by frame from beginning to end. However, the plots themselves may be non-linear, employing flashbacks and parallel subplots. Maps have a spatial structure. Items are placed according to physical proximity, although the most useful maps cheat, sacrificing accuracy for clarity.

The structure of information defines the primary ways in which users can navigate. Major organization structures that apply to web site and intranet architectures include the hierarchy, the database-oriented model, and hypertext.

The hierarchy: A top-down approach

The foundation of almost all good information architectures is a well-designed hierarchy. In this hypertextual world of nets and webs, such a statement may seem blasphemous, but it's true. The mutually exclusive subdivisions and parent-child relationships of hierarchies are simple and familiar. We have organized information into hierarchies since the beginning of time. Family trees are hierarchical. The top-down approach allows you to quickly get a handle on the scope of the web site without going through an extensive content inventory process.

element of a simple hierarchical model

```
  Plants
   /   \
  /     \
Flowers Trees
  /     \
Annals  Pteridales  Conifers  Deciduous
```
Designing hierarchies

When designing information hierarchies on the Web, you should remember a few rules of thumb. First, you should be aware of, but not bound by, the idea that hierarchical categories should be mutually exclusive. Within a single organization scheme, you will need to balance the tension between exclusivity and inclusivity. Ambiguous organization schemes in particular make it challenging to divide content into mutually exclusive categories.

Second, it is important to consider the balance between breadth and depth in your information hierarchy. Breadth refers to the number of options at each level of the hierarchy. Depth refers to the number of levels in the hierarchy. If a hierarchy is too narrow and deep, users have to click through an inordinate number of levels to find what they are looking for.

Hypertext

Hypertext is a relatively new and highly nonlinear way of structuring information. A hypertext system involves two primary types of components: the items or chunks of information which are to be linked, and the links between those chunks. These components can form hypermedia systems that connect text, data, image, video, and infoarch chunks. Hypertext chunks can be connected hierarchically, non-hierarchically, or both.

The relational database model: A bottom-up approach

Most of us are familiar with databases. In fact, our names, addresses, and other personal information are included in more databases than we care to imagine. A database is a collection of records. Each record has a number of associated fields. The database model has limitations. The records must follow rigid rules. Within a particular record type, each record must have the same fields, and within each field, the formatting rules must be applied consistently across records. This highly structured approach does not work well with the heterogeneous content of many web sites. Also, technically it's not easy to place the entire contents (including text, graphics, and hypertext links) of every HTML page into a database. Such an approach can be very expensive and time consuming. For these reasons, the database model is best applied to subsites or collections of structured, homogeneous information within a broader web site. For example, staff...
Subject: web engineering

directories, news release archives, and product catalogs are excellent candidates for the database model.

Figure 3-13. This entity relationship diagram (ERD) shows a structured approach to database design see that entities (e.g., Resource) have attributes (e.g., Name, URL). Ultimately, entities and attributes become records and fields in the database. An ERD also shows relationships between entities. For example, we see that each resource is available at one or more locations. The ERD is used to visualize and refine the data model, before design and population of the database..
Subject: web engineering

Lecture 5


Creating Cohesive Organization Systems

You also need to think about the organization structures that influence how users can navigate through these schemes. Should you use a hierarchy or would a more structured database-model work best? Perhaps a loose hypertextual web would allow the most flexibility? Taken together, in the context of a large web site development project, these questions can be overwhelming. That's why it's important to break down the site into its components, so you can tackle one question at a time. Also, keep in mind that all information retrieval systems work best when applied to narrow domains of homogeneous content. By decomposing the content collection into these narrow domains, you can identify opportunities for highly effective organization systems.

In considering which organization schemes to use, remember the distinction between exact and ambiguous schemes. Exact schemes are best for known-item searching, when users know precisely what they are looking for. Ambiguous schemes are best for browsing and associative learning, when users have a vaguely defined information need. Whenever possible, use both types of schemes. Also, be aware of the challenges of organizing information on the Web. Language is ambiguous, content is heterogeneous, people have different perspectives, and politics can rear its ugly head. Providing multiple ways to access the same information can help to deal with all of these challenges. When thinking about which organization structures to use, keep in mind that large web sites and intranets typically require all three types of structure. The top-level, umbrella architecture for the site will almost certainly be hierarchical. As you are designing this hierarchy, keep a lookout for collections of structured, homogeneous information. These potential subsites are excellent candidates for the database model. Finally, remember that less structured, creative relationships between content items can be handled through hypertext. In this way, all three organization structures together can create a cohesive organization system.

Designing Navigation Systems

On the Web, navigation is rarely a life or death issue. However, getting lost in a large web site can be confusing and frustrating. While a well-designed hierarchical organization scheme will reduce the likelihood that users will become lost, a complementary navigation system is often needed to provide context and to allow for greater flexibility of movement within the site. Navigation systems can be designed to support associative learning by featuring resources that are related to the content currently being displayed. For example, a page that describes a product may include see also links to related products and services (this
Subject: web engineering

type of navigation can also support a company's marketing goals). As users move through a well-designed navigation system, they learn about products, services, or topics associated to the specific content they set out to find.
Any page on a web site may have numerous opportunities for interesting connections to other areas of the site. The constant challenge in navigation system design is to balance this flexibility of movement with the danger of overwhelming the user with too many options.
Navigation systems are composed of a variety of elements. Some, such as graphical navigation bars and pop-up menus, are implemented on the content-bearing pages themselves. Others, such as tables of contents and site maps, provide remote access to content within the organization structure. While these elements may be implemented on each page, together they make up a navigation system that has important site-wide implications. A well-designed navigation system is a critical factor in determining the success of your web site.

Browser Navigation Features
When designing a navigation system, it is important to consider the environment the system will exist in. On the Web, people use web browsers such as Netscape Navigator and Microsoft Internet Explorer to move around and view web sites. These browsers sport many built-in navigation features.
Open URL allows direct access to any page on a web site. Back and Forward provide a bidirectional backtracking capability. The History menu allows random access to pages visited during the current session, and Bookmark enables users to save the location of specific pages for future reference. If the hypertext link leads to another web site on another server, prospective view provides the user with basic information about this off-site destination.

Building context:
With all navigation systems, before we can plot our course, we must locate our position. Whether we're visiting Yellowstone National Park or the Mall of America, the You Are Here mark on fixed-location maps is a familiar and valuable tool. Without that landmark, we must struggle to triangulate our current position using less dependable features such as street signs or nearby stores. In designing complex web sites, it is particularly important to provide context within the greater whole. always follow a few rules of thumb to ensure that your sites provide contextual clues. First, all pages should include the organization's name. This might be done as part of the title or header of the page. As a user moves through the levels of a site, it should be clear that they are still within that site. Carrying the graphic identity throughout the site supports such context and consistency. In addition, if a user bypasses the front door and directly accesses a subsidiary page of the site, it should be clear which site he or she is on. Second, the navigation system should present the structure of the information hierarchy in a clear and consistent manner and indicate the location within that hierarchy.
Lecture 6

Topic: Types of Navigation systems

Types of Navigation Systems

A complex web site often includes several types of navigation systems. To design a successful site, it is essential to understand the types of systems and how they work together to provide flexibility and context.

Hierarchical Navigation Systems

Although we may not typically think of it this way, the information hierarchy is the primary navigation system. From the main page to the destination pages that house the actual content, the main options on each page are taken directly from the hierarchy. The hierarchy is extremely important, but also rather limiting. It is these limitations that often require additional navigation systems.

Global Navigation Systems

A global or site-wide navigation system often complements the information hierarchy by enabling greater vertical and lateral movement throughout the entire site. At the heart of most global navigation systems are some standard rules that dictate the implementation of the system at each level of the site.

The simplest global navigation system might consist of a graphical navigation bar at the bottom of each page on the site. On the main page, the bar might be unnecessary, since it would duplicate the primary options already listed on that page. On second level pages, the bar might include a link back to the home page and a link to the feedback facility.

Local Navigation Systems

For a more complex web site, it may be necessary to complement the global navigation system with one or more local navigation systems. To understand the need for local navigation systems, it is necessary to understand the concept of a sub-site.

For example, a software company may provide an online product catalog as one area in their web site. This product catalog constitutes a sub-site within the larger web site of the software company. Within this sub-site area, it makes sense to provide navigation options unique to the product catalog, such as browsing products by name or format or market.
Ad Hoc Navigation

Relationships between content items do not always fit neatly into the categories of hierarchical, global, and local navigation. An additional category of *ad hoc* links is more editorial than architectural. Typically an editor or content specialist will determine appropriate places for these types of links once the content has been placed into the architectural framework of the web site. In practice, this usually involves representing words or phrases within sentences or paragraphs (i.e., prose) as embedded hypertext links. This approach can be problematic if these ad hoc links are important, since usability testing shows "a strong negative correlation between embedded links (those surrounded by text) and user success in finding information." Apparently, users tend to scan pages so quickly that they often miss these less conspicuous links. You can replace or complement the embedded link approach with external links that are easier for the user to see.
Subject: web engineering

Lecture 7


Integrated Navigation Elements

In global and local navigation systems, the most common and important navigation elements are those that are integrated into the content-bearing pages of the web site. As users move through the site or sub-site, these are the elements they see and use again and again. Most integrated navigation elements fit into one of two categories: navigation bars and pull-down menus.

Navigation Bars:
You can implement navigation bars in many ways and use them for the hierarchical, global, and local navigation systems. In simplest form, a navigation bar is a collection of hypertext links grouped together on a page. Alternatively, the navigation bar may be graphical in nature, implemented as an image map or as graphic images within a table structure.
Graphical navigation bars may employ several techniques for conveying content and context, including textual labels and icons. Textual labels are the easiest to create and by far most clearly indicate the contents of each option. Icons, on the other hand, are relatively difficult to create and often fail to indicate the contents of each option. Graphic navigation bars tend to look nicer but can significantly slow down the page loading speed.

Frames:

Frames present an additional factor to consider in the application of textual or graphical navigation bars. Frames allow you to define one or more independently scrollable "panes" within a single browser window. Hypertextual links within one pane can control the content displayed in other panes within that same window. This enables the designer to create a static or independently scrolling navigation bar that appears on every page in that area of the web site.

However, frames present several serious problems, both from the consumer's and producer's perspective. Architects should proceed very carefully in considering framebased navigation solutions. Let's review a few of the major considerations:

- Screen real estate
- The page model
- Display speed
- Complex design
- Pull-Down Menus
Remote Navigation Elements

remote navigation elements are similar to software documentation or help systems. Documentation can be very useful but will never save a bad product. Instead, remote navigation elements should be used to complement a solid internal organization and navigation system. You should provide them but never rely on them. While remote navigation elements can enhance access to web site content by providing complementary ways of navigating, they should not be used as replacements or bandages for poor organization and navigation systems.

The Table of Contents

The table of contents and the index are the state of the art in print navigation. In a book or 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. Similarly, the table of contents for a web site presents the top few levels of the hierarchy. It provides a broad view of the content in the site and facilitates random access to segmented portions of that content. A web-based table of contents can employ hypertext links to provide the user with direct access to pages of the site.

The design of a table of contents significantly affects its usability. When working with a graphic designer, make sure he or she understands the following rules of thumb:

- Reinforce the information hierarchy so the user becomes increasingly familiar with how the content is organized.
- Facilitate fast, direct access to the contents of the site for those users who know what they want.
- Avoid overwhelming the user with too much information. The goal is to help, not scare, the user.

The Index: In selecting items for the index, keep in mind that an index should point only to destination pages, not navigation pages. Navigation pages help users find (destination) pages through the use of menus that begin on the main page and descend through the hierarchy.

The Site Map: A real site map presents the information architecture in a way that goes beyond textual representation.

The Guided Tour: A guided tour serves as a nice tool for introducing new users to the major content areas of a web site. It can be particularly important for restricted access web sites (such as online magazines that charge subscription fees) because you need to show potential customers what they will get for their money.

A guided tour should feature linear navigation (new users want to be guided, not thrown in), but a hypertextual navigation bar may be used to provide additional flexibility. The
Subject: web engineering

tour should combine screenshots of major pages with narrative text that explains what can be found in each area of the web site

Lecture 8


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 dynamic 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 challenges 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 web site will determine design of the global navigation system. Based on the hierarchy, you will be able to 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, your design of the higher-order navigation system will influence decisions at the next level.

Once you've designed the integrated navigation system, you can consider the addition of one or more remote navigation elements. In most cases, you will need to choose between a table of contents, an index, and a site map. 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 itself to visualization? If so, a site map may be appropriate. Is there a need to help new or prospective users to understand what they can do with the 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 refine an elegant navigation system that really works.

Searching Systems
Searching and Your Web Site

When Not To Make Your Site Searchable: Your site should of course support the finding of its information. But don't assume a search engine alone will satisfy all users' information needs. While many users want to search a site, some just want to browse it. Because many site developers see search engines as the solution to the problems that users are experiencing when trying to find information in their sites, search engines become bandages for sites with poorly designed browsing systems.

Search engines are fairly easy to get up and running, but like much of the Web, they are difficult to set up effectively. As a user of the Web, you've certainly seen incomprehensible search interfaces, and we're sure that your queries have retrieved some pretty strange results. This often is the result of a lack of planning by the site developer, who probably installed the search engine with its default settings, pointed it at his or her site, and forgot about it. So, if you don't plan on putting some significant time into configuring your search engine properly, reconsider your decision to implement it. Now that we've got our warnings and threats out of the way, we'll discuss when to implement searching systems, and how you can make them work better.

When To Make Your Site Searchable

Most web sites, as we know, aren't planned out in much detail before they're built. Instead, they grow organically. This may be all right for smaller web sites that aren't likely to expand much, but for ones that become popular, more and more content and functional features get added haphazardly, leading to a navigation nightmare. Your site probably doesn't contain as much content as Yahoo! does, but if it's a substantial site, it probably merits a search engine. There are good reasons for this: users won't be willing to browse through your site's structure. Their time is limited, and their cognitive overload threshold is lower than you think. Interestingly, sometimes users won't browse for the wrong reasons; that is, they search when they don't necessarily know what to search for. Even though they would be better served by browsing, they search anyway. You should also consider creating a searching system for your site if it contains highly dynamic content. For example, if your site is a Web-based newspaper, you could be adding dozens of story files daily. For this reason, you probably wouldn't have the time each day to maintain elaborate tables of contents, browsable indices, and other browsing systems. A search engine can help you by automatically indexing the contents of the site once or many times per day. Automating this process ensures that users have quality access to your site's content, and you can spend time doing things other than manually indexing and linking the story files.

Understanding How Users Search
Subject: web engineering

Assuming you've decided to implement a searching system for your web site, it's important to understand how users really search before designing it. We'll try to condense decades of research and experience generated by the field of information retrieval into the next few paragraphs. But it really boils down to this point: searching systems can and should vary as much as browsing systems or any other components of web sites do, because all users aren't alike, and information retrieval is much harder than most people realize.

Users Have Different Kinds of Information Needs

**Known-item searching:** users' information needs are clearly defined and have a single, correct answer

**Existence searching:** some users know what they want but don't know how to describe it or whether the answer exists at all

**Exploratory searching:** Some users know how to phrase their question, but don't know exactly what they're hoping to find, and are really just exploring and trying to learn more.

**Comprehensive searching (research):** 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 into this category.
Lecture 9

Topic: Designing the Search Interface

Designing the Search Interface

Assuming that a search facility is needed, a designer should first and foremost consider what the user wants to search for. Far too often, search engines are added to a site and set to index everything using a free text search. Similar to a Web-wide search, users pound their heads as they search for a particular part number like KF-456 only to be shown every single document the part number occurs in, ranging from press releases to technical notes. To the user, the ordering of the documents from this type of search may seem arbitrary, with the most important document not appearing first in the list. What's interesting is why this form of search was used. Designers assume that since public search engines work like this, so should their local search engine. This seems like a good idea—users are familiar with formulating search strings at public sites and bring this knowledge with them to your site. However, global search engines are not very accurate for a variety of reasons, including the fact that numerous sites try to fight their way to the top of returned results. Public search engine results don't always seem to make sense, and the ordering often seems more random than systematic.

Consider that in your own site, if you want a particular page to be shown when a user types in "Robot Butler," you can cause that page to be shown. Remember, when building a local search facility, to copy the style, syntax, and interface of public Web search engines, but don't imitate their imprecise functionality.

The main advantage of local searching is that you can utilize controlled vocabularies to deal with what users will probably want to search for. Besides relating keywords with certain pages in a more precise manner, you may even suggest common queries for users to run. Remember, local search engines provide designers with a much greater degree of control than public search engines.

basic points should be considered when designing a search interface

Support Different Modes of Searching

Before diving into design, think hard about why users are searching your site, and what they want to get out of their search. Are they likely to search for certain types of information, such as specific product descriptions or staff directory entries?
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.

**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. This advice may seem painfully obvious, but because many search engines are packaged as ready-to-go add-ons to a site, site developers don't bother to customize them.

**Search Options Should Be Clear**

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 text 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? 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 such 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's important to provide the user with some tricks to handle the following three situations:
  - a. "I'm getting too much stuff."
  - b. "I'm not getting anything."
  - c. "The stuff I'm getting stinks!"

**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.

**Display Search Results Sensibly**

You can configure how your search engine displays search results in many ways. There is no right way to do it. How you configure your search engine's results depends on two factors.

The first factor is the degree of structure your content has. The other factor is what your site's users really want.
Subject: web engineering

**More About Relevance**

How relevant the results displayed are.
Always Provide the User with Feedback
When a user executes a search, he or she expects results. Usually, a query will retrieve 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 specially for these cases

**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's title; this will make it easier for users to find it in their browser's history lists.

Let the user know how many documents 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 retrieval 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 start 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.
Lecture 10

Topic: Indexing the Right Stuff, To search or Not To Search, Grouping Content

Indexing the Right Stuff:

Let's assume that you do intend to slap a search engine on top of your web site. Shouldn't be a problem right? Just point the indexer at the directory where all the pages live. Of course, you knew it wasn't that simple. 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.

Indexing the Entire Site

Search engines are frequently used to index an entire site without regard for the content and how it might vary -- every word of every page, whether it contains real content or help information, advertising, navigation menus, and so on. However, searching works much better when the information space is defined narrowly and contains homogeneous content. In other words, the more you search through indices that combine *apples and oranges*, the worse your retrieval results will be.

Search Zones: Selectively Indexing the Right Content

Search zones are subsets of a web site that have been indexed separately from the rest of the site's content. 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 the result is improved retrieval performance. The user is simply less likely to retrieve irrelevant information.

You can create search zones in many ways. Examples of four common approaches are:

- by content type
- by audience
- by subject
- by date

indexing similar content types

Most web sites contain, at minimum, two major and dissimilar types of pages: *navigation* and *destination*. Destination pages contain the actual information you want from a web site: sport scores, book reviews, software documentation, and so on. The primary purpose of a site's navigation pages is to *get you to the destination pages*. 
Navigation pages may include main pages, search pages, and pages that help you browse a site.

Indexing for specific audiences

If you've already decided to create an architecture for your site that uses an audience-oriented organization scheme, it may make sense to create search zones by audience breakdown as well.

Indexing by subject

If your site uses a strong subject-oriented or topical organization scheme, you've already distinguished many of the site's search zones.

Indexing recent content

Chronologically organized content allows for perhaps the easiest implementation of search zones. Because dated materials are generally not ambiguous, indexing them by date is straightforward.

Grouping content

grouping content into the top-level categories of an information hierarchy is typically the most important and challenging process. 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? (explained in topic: organizing information)
Lecture 11

Topic: High-Level Architecture Blueprints

High-Level Architecture Blueprints

The collaborative brainstorming process is exciting, chaotic, and fun. However, sooner or later, you must hole up away from the crowd and transform this chaos into order. Blueprints are the architect's tool of choice for performing this transformation. The very act of shaping ideas into the more formal structure of a blueprint forces you to become realistic and practical. If brainstorming takes you to the top of the mountain, blueprinting brings you back down to reality. Ideas that seemed brilliant on the white board may not pan out when you attempt to organize them in a practical manner. It's easy to throw around concepts such as audience-specific gateways and adaptive information architectures. It's not so easy to define on paper exactly how these concepts will be applied to a specific web site.

During the conceptual 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 web site. This exploration may involve several iterations as you further define the information architecture. High-level blueprints are great for stimulating discussions focused on the organization and management of content as well as the desired access pathways for users. These blueprints can be created by hand, but we prefer to use diagramming software such as Visio or NetObjects Fusion. These products not only help you to quickly layout your architecture blueprints, but can also help with site production and maintenance.
This high-level blueprint shows pages, components within pages, groups of pages, and relationships between pages. The grouping of pages can inform page layout. For example, the three value-added guides should be presented together, whereas Search & Browse, Feedback, and News should be presented separately.

Moving up from the sub-sites, we see a directory of sub-site records. This directory serves as a card catalog that provides easy access to the sub-sites. There is a sub-site record for each sub-site. Each record consists of fields such as title, description, keywords, audience, format, and topic that describe the contents of that sub-site. By creating a standardized record for each sub-site, we are actually creating a database of sub-site records. This database approach enables powerful known-item searching and more exploratory browsing. As you can see from the Search & Browse page, users can search and browse by title, audience, format, and topic.

We also see three value-added guides. These guides take the form of simple narratives or stories that introduce new users to the organization and to the web site. Interwoven throughout the text of these narratives are in-context links to selected sub-sites. They guide users through the site in an interesting and friendly way.

Finally, we see a dynamic news billboard (perhaps implemented through Java or JavaScript) that rotates the display of featured news headlines and announcements. In addition to bringing some action to the main page, this billboard provides yet another way to access important content that might otherwise be buried within a sub-site.

You should note that these high-level blueprints leave out quite a bit of information. They focus on the major areas of the site, ignoring navigation elements and page-level details. These omissions are by design, not by accident.
Subject: web engineering

Detailed page-level blueprints come later in the process.
Subject: web engineering

Lecture 12

Topic: Architectural Page Mockups, Design Sketches

Architectural Page Mockups

Information architecture blueprints are most useful for presenting a bird's-eye view of the web site. However, they do 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 does a perfect job of conveying all aspects of an information architecture to all audiences. Because information architectures are multidimensional, it's important to show them in multiple ways.

For these reasons, architectural page mockups are useful tools during conceptual design for complementing the blueprint view of the site. Mockups are quick and dirty textual documents that show the content and links of major pages on the web site. 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 the design of subsidiary pages.
In this architectural mockup of a combination search/browse page, we show an area for entering queries and an area for browsing. We typically use a word processor like Microsoft Word to create these mockups quickly. However, you can also create quick and dirty HTML mockups, and even work quite interactively with the graphic designer.

In the example in , you see that mockups are easier to read than blueprints. By integrating aspects of the organization, labeling, and navigation systems into 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. In this example, the search interface and the browsing options are two separate content groups. You can also indicate prominence in these mockups. Placing a content group at the top of the page or using a larger font size indicate 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.

**Design Sketches**

Once you've developed high-level blueprints and architectural page mockups, you're ready to collaborate with your graphic designer to create design 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.
Subject: web engineering

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 is a wonderful opportunity for interdisciplinary user interface design.

Using the architectural mockups as a guide, the designer begins sketching pages 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-away table of contents feature? We've had pretty good reactions to that type of approach from 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.

As you can see, the design of these sketches requires the involvement of people from all three teams. It is much cheaper and easier for the group to work with the designer on these rough sketches than to begin with actual HTML page layouts and graphics. These sketches allow rapid iteration and intense collaboration. The final product of a sketching session might look something like that in Figure
In this example, Employee Handbook, Library, and News are grouped together as the major areas of the web site. Search/Browse and Guidelines/Policies make up the bottom of the page navigation bar. A news area defines space for a dynamic Java-based news panel.
Lecture 13

Topic: HTML Basic Concepts

HTML basic concepts:

What is HTML?
HTML is a language for describing web pages.
HTML stands for Hyper Text Markup Language
HTML is not a programming language, it is a markup language
A markup language is a set of markup tags
HTML uses markup tags to describe web pages

HTML Tags
HTML markup tags are usually called HTML tags
HTML tags are keywords surrounded by angle brackets like <html>
HTML tags normally come in pairs like <b> and </b>
The first tag in a pair is the start tag, the second tag is the end tag
Start and end tags are also called opening tags and closing tags

HTML Documents = Web Pages
HTML documents describe web pages
HTML documents contain HTML tags and plain text
HTML documents are also called web pages
The purpose of a web browser (like Internet Explorer or Firefox) is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page:

<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>

The text between <html> and </html> describes the web page
The text between <body> and </body> is the visible page content
The text between <h1> and </h1> is displayed as a heading
The text between <p> and </p> is displayed as a paragraph
HTML documents are defined by HTML elements.
Subject: web engineering

HTML Elements

An HTML element is everything from the start tag to the end tag:

<table>
<thead>
<tr>
<th>Start tag *</th>
<th>Element content</th>
<th>End tag *</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;p&gt;</td>
<td>This is a paragraph</td>
<td>&lt;/p&gt;</td>
</tr>
<tr>
<td>&lt;a href=&quot;default.htm&quot;&gt;</td>
<td>This is a link</td>
<td>&lt;/a&gt;</td>
</tr>
<tr>
<td>&lt;br /&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* The start tag is often called the opening tag. The end tag is often called the closing 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
- Nested HTML Elements
- Most HTML elements can be nested (can contain other HTML elements).

HTML Document Example

```html
<html>
<body>
<p>This is my first paragraph.</p>
</body>
</html>
```

The example above contains 3 HTML elements.

The <p> element:
- <p>This is my first paragraph.</p>
The <p> element defines a paragraph in the HTML document.
The element has a start tag <p> and an end tag </p>.

The element content is: This is my first paragraph.

The <body> element:
- <body>
- <p>This is my first paragraph.</p>
- </body>

Subject: web engineering

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 p element).

Empty HTML Elements

HTML elements with no content are called empty elements. <br> is an empty element without a closing tag (the <br> tag defines a line break).

HTML Attributes

HTML elements can have attributes. Attributes provide additional information about an element. Attributes are always specified in the start tag. Attributes come in name/value pairs like: name="value"

Attribute Example

HTML links are defined with the <a> tag. The link address is specified in the href attribute:

<a href="http://www.google.com">This is a link</a>

HTML Headings

Headings are defined with the <h1> to <h6> tags. <h1> defines the most important heading. <h6> defines the least important heading.

<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>

The HTML Style Attribute

The purpose of the style attribute is:
To provide a common way to style all HTML elements. Styles was introduced with HTML 4, as the new and preferred way to style HTML elements. With HTML styles, styles can be added to HTML elements directly by using the style attribute, or indirectly in separate style sheets (CSS files).

HTML Style Example - Background Color

The background-color property defines the background color for an element:

Example

<html>

<body style="background-color: yellow">
<h2 style="background-color: red">This is a heading</h2>

</body>
</html>
Subject: web engineering

<html>
<p style="background-color:green">This is a paragraph.</p>
</html>

HTML Tables

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc.

Table Example

```html
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How the HTML code above looks in a browser:

```
+----------+----------+
| row 1, cell 1 | row 1, cell 2 |
+----------+----------+
| row 2, cell 1 | row 2, cell 2 |
+----------+----------+
```
Subject: web engineering

Lecture 14

Topic: Good Web Design, Process of Web Publishing

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

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

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

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

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

The process of designing web pages, web sites, web applications or multimedia for the Web may utilize multiple disciplines, such as animation, authoring, communication.
Subject: web engineering

design, corporate identity, graphic design, human-computer interaction, information architecture, interaction design, marketing, photography, search engine optimization and typography.

One of the elements of **good web design** is a lack of the elements that make bad web design. In addition, keep these concepts in mind:

**Text**
- Background does not interrupt the text
- Text is big enough to read, but not too big
- The hierarchy of information is perfectly clear
- Columns of text are narrower than in a book to make reading easier on the screen

**Navigation**
- Navigation buttons and bars are easy to understand and use
- Navigation is consistent throughout the site
- Navigation buttons and bars provide the visitor with a clue as to where they are, what page of the site they are currently on
- Frames, if used, are not obtrusive
- A large site has an index or site map

**Links**
- Link colors coordinate with page colors
- Links are underlined so they are instantly clear to the visitor

**Graphics**
- Buttons are not big and dorky
- Every graphic has an alt label
- Every graphic link has a matching text link
- Graphics and backgrounds use browser-safe colors
- Animated graphics turn off by themselves

**General Design**
- Pages download quickly
- First page and home page fit into 800 x 600 pixel space
- All of the other pages have the immediate visual impact within 800 x 600 pixels
- Good use of graphic elements (photos, subheads, pull quotes) to break up large areas of text
- Every web page in the site looks like it belongs to the same site; there are repetitive elements that carry throughout the pages

**Publishing** is the process of production and dissemination of literature or information - the activity of making information available for public view. In some cases authors may
Subject: web engineering

be their own publishers, meaning: originators and developers of content also provide media to deliver and display the content.

Plan Your Web Site
Determine the audience, purpose and content for your Web site. Who is the audience? What are they looking for on your site? How can you organize your content so the audience can find what they want? Do you have information that they might not realize they will need? Do you need to lay out a process, or sequence of steps? Try to do this from the perspective of your audience, rather than from your own organizational structure, reporting needs, or processes.
Think through the organization of your content to present your information in a logical manner, understandable to your anticipated audience. Prepare a schedule for maintaining links and updating content.

Get Approval (new sites or additions to existing sites)

Work through your normal reporting channels for approval to produce a Web site for your class, program, department or unit

Avoid Duplication
Evaluate the information already published Web site to avoid duplication of information or contradictions with prior publications. If you have doubts about whether your content may already exist on the Web site, check with Web Coordinator. Do not post duplicate copies of existing documents; link to them at their original Web address. This helps prevent broken links.
Official institutional publications are already on the Web. These include the undergraduate Bulletin, the Graduate Bulletin, handbooks and so on. You must link to these documents to avoid multiple or conflicting versions of University information.
If your site includes information about programs, minors, specializations or course descriptions, you must call them by their official name and link to existing documents.
Lecture 15

Topic: Phases of Web Site development, Structure of HTML documents

The Web Site Design and Development Process

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.

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

Phase One: Information Gathering

The first step in designing a successful web site is to gather information.
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?
**Goals**
What do you hope to accomplish by building this web site?
**Target Audience**
Is there a specific group of people that will help you reach your goals?
**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…?
**Phase Two: 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. 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.

**Phase Three: Design**

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. 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. 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.

**Phase 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.
Subject: web engineering

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.

Phase 5: testing and delivery

At this point, 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.

The basic technologies currently used are XHTML and CSS (Cascading Style Sheets). As part of testing, designer should check to be sure that all of the code written for 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.

Phase 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. update your own content, there is something called a CMS (Content Management System) that can be implemented to the 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!

Structure of HTML documents

A web page constructed using HTML has a basic and essential structure. The page always begins with the start tag of the html element and always terminates with the end tag of the html element as follows:
Subject: web engineering

<html>

...web page...

</html>

The html element basically tells your computer that this is an HTML document. All other element tags are 'nested' within the start and end html tags. The web page is then further subdivided into two main sections which are the 'head' and the 'body'.

The head section begins with the <head> start tag and terminates with the </head> end tag. Immediately following this comes the <body> start tag and just before the html end tag comes the </body> end tag.

There is only one set of <html>...</html> tags, one set of <head>...</head> tags and one set of <body>...</body> tags. This basic HTML web page structure can be illustrated by the following example:

<html>
<body>
</body>
</html>

The head section or document head has little content and mostly contains HTML coded instructions on how to title, categorize and 'run' the web page. The body section or document body on the other hand contains almost all of the content that you will put on your web page and this content —usually text but can also be pictures and sounds— is formatted using more HTML code. All text that you place outside of any angle brackets will become 'visible text' and will be displayed by your web browser on your web page. By placing that text in between the start and end tags of certain HTML elements, you can instruct a web browser where and how to display that text.

Adding a Title to your Web Page

Web pages usually have a title that appears in the title bar that runs across the very top of the web page. This title is created using the <title>...</title> tags which are themselves always nested within the <head>...</head> tags. All text appearing after the <title> start tag and before the </title> end tag will be displayed as your web page title. Hence the following HTML code will produce a web page entitled 'My Home Page':

<html>
<head>
<title> My Home Page </title>
</head>
</html>
Adding Content to your Web Page

Now to add some content to your web page all you have to do is type some text in between the `<body>` tags. So let's, for example, put the words 'HELLO WORLD!' on your web page.

```html
<html>
<head>
<title>My Home Page</title>
</head>
<body>
HELLO WORLD!
</body>
</html>
```

This will produce a web page that looks like this:

Of course, you're probably going to want to do a lot more than just create a web page with a title and some plain text on a blank white background.
Subject: web engineering

Lecture 16

Topic: HTML Elements-Core attributes

HTML Elements

An HTML element is everything from the start tag to the end tag:

<table>
<thead>
<tr>
<th>Start tag *</th>
<th>Element content</th>
<th>End tag *</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;p&gt;</td>
<td>This is a paragraph</td>
<td>&lt;/p&gt;</td>
</tr>
<tr>
<td>&lt;a href=&quot;default.htm&quot; &gt;</td>
<td>This is a link</td>
<td>&lt;/a&gt;</td>
</tr>
<tr>
<td>&lt;br /&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* The start tag is often called the opening tag. The end tag is often called the closing 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

Nested HTML Elements

Most HTML elements can be nested (can contain other HTML elements). HTML documents consist of nested HTML elements.

HTML Document Example

```html
<html>
<body>
<p>This is my first paragraph.</p>
</body>
</html>
```

The example above contains 3 HTML elements.
Subject: web engineering

The `<p>` element:

```html
<p>This is my first paragraph.</p>
```

The `<p>` element defines a paragraph in the HTML document. The element has a start tag `<p>` and an end tag `</p>`. The element content is: This is my first paragraph.

**The `<body>` element:**

```html
<body>
<p>This is my first paragraph.</p>
</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 (an `<p>` element).

**The `<html>` element:**

```html
<html>
<body>
<p>This is my first paragraph.</p>
</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 element).

**Some HTML elements will be displayed correctly even if you forget the end tag:**

```html
<p>This is a paragraph
<p>This is a paragraph
```

The example above will work in most browsers, but don't rely on it. Forgetting the end tag can produce unexpected results or errors.

**Empty HTML Elements**

HTML elements with no content are called empty elements.

```html
<br>
```

is an empty element without a closing tag (the `<br>` tag defines a line break). Tip: In XHTML, all elements must be closed. Adding a slash inside the start tag, like
Subject: web engineering

<br />, is the proper way of closing empty elements in XHTML

HTML Tip: Use Lowercase Tags
HTML tags are not case sensitive: <p> means the same as <P>. Many web sites use uppercase HTML tags.
Subject: web engineering

**HTML Attributes**

HTML elements can have **attributes**. Attributes provide **additional information** about an element. Attributes are always specified in the **start tag**. Attributes come in name/value pairs like: `name="value"`

**Always Quote Attribute Values**

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed. Below is a list of some attributes that are standard for most HTML elements:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>classname</td>
<td>Specifies a classname for an element</td>
</tr>
<tr>
<td>id</td>
<td>id</td>
<td>Specifies a unique id for an element</td>
</tr>
<tr>
<td>style</td>
<td>style_definition</td>
<td>Specifies an inline style for an element</td>
</tr>
<tr>
<td>title</td>
<td>tooltip_text</td>
<td>Specifies extra information about an element (displayed as a tool tip)</td>
</tr>
</tbody>
</table>

**Core Attributes**

**ID**

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 and may be followed by letters (A-Za-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:

```html
<P ID=firstp>My first paragraph.</P>
<P ID=secondp>My second paragraph.</P>
```

**CLASS**

The **CLASS** attribute specifies the element to be a member of one or more classes. Classes allow authors to define specific *kinds* of a given element. For example, an author could use `<CODE CLASS=Java>` when giving Java code and `<CODE CLASS=Perl>` when giving Perl code.
Subject: web engineering

Unlike with the ID attribute, any number of elements can share the same class. An element may also belong to multiple classes; the CLASS attribute value is a space-separated list of class names. The value is case-sensitive.

Use of the class attribute in an HTML document:
<html>
<head>
  <style type="text/css">
    h1.intro {color:blue;}
    p.important {color:green;}
  </style>
</head>

<body>
  <h1 class="intro">Header 1</h1>
  <p>A paragraph.</p>
  <p class="important">Note that this is an important paragraph.</p>
</body>
</html>

STYLE

The STYLE attribute allows authors to specify style rules inline for a single occurrence of an element. An example follows:

    <p>A popular font for on-screen reading is <span style="font-family: Verdana">Verdana</span>.</p>

TITLE

Syntax

<element title="value"/>

The TITLE attribute provides a title for an element and is commonly implemented as a "tooltip" on visual browsers. The attribute is most useful with A, AREA, LINK, and IMG elements, where it provides a title for the linked or embedded resource.

Use of the title attribute in an HTML document:

    <abbr title="World Health Organization">WHO</abbr> was founded in 1948.
    <p title="free journal ">google.com</p>
Lecture 17

Topic: Language attributes

Language Attributes

Not valid in base, br, frame, frameset, hr, iframe, param, and script elements.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dir</td>
<td>ltr</td>
<td>Specifies the text direction for the content in an element</td>
</tr>
<tr>
<td></td>
<td>rtl</td>
<td></td>
</tr>
<tr>
<td>lang</td>
<td>language_code</td>
<td>Specifies a language code for the content in an element.</td>
</tr>
<tr>
<td>xml:lang</td>
<td>language_code</td>
<td>Specifies a language code for the content in an element, in XHTML documents.</td>
</tr>
</tbody>
</table>

Dir

This attribute is used to indicate the directionality of the flow of the content for the current element. This becomes most helpful in bi-directional language scenarios where intrinsic dimension may be ambiguous. On block elements, this attribute indicates the base directionality of the text in the block. For inline elements this attribute starts a new embedding level for direction-dependent content. If this attribute is omitted for an inline element, a new embedding level is not created. If used as an attribute of the HTML tag, it will apply to the entire document. When used with other tags, it only applies to the text under the influence of that tag. For example, when used with the table tag, the first column of the table will start on the right side and each additional column will be farther to the left.

dir="ltr"
The ltr value, which directs the text to flow from left to right, is the default.

dir="rtl"
The rtl value directs the text to flow from right to left.

Lang

The HTML lang attribute can be used to declare the language of a Web page or a portion of a Web page. This is meant to assist search engines and browsers.
Subject: web engineering

This attribute is used to specify the language of the enclosed content. This property can be useful in several ways - it can be used to ensure proper display of language-specific character usage (such as quotes or decimal points), for speech synthesis, search engine content classification or clarification of ambiguous character usage.

This attribute takes as its value a string that identifies a language system used for communication (with the exception of computer languages.). A language tag is composed of one or more parts: A primary language tag and a possibly empty series of subtags:

language-tag= [Primary Language Tag] ("-" [Language Subtag])*  
[Primary Language Tag] = "i" (for IANA defined languages) | "x" (custom/private use language) | [ISO 639 2-letter Language Code]  
[Language Subtag] = [ISO 3166 2-letter country code] | [dialect or other locale/situation specific language]  
Language tags are case-insensitive and spaces are not allowed.

The LANG attribute overrides any language value specified by any parent elements of the current element. If no value is specified at any of these levels, the language inheritance mechanism goes up to the HTTP protocol header 'Content-Language.' If this is also not specified, a default may be determined from the user's browser settings or some other criteria.

According to the W3C recommendation you should declare the primary language for each Web page with the lang attribute inside the <html> tag, like this:

<html lang="en">
</html>

Xml-lang

xml:lang="language code"

The xml:lang attribute is identical in its usage and purpose to the lang attribute, but with one difference: it must be applied within documents that use an XHTML doctype. It may be that this attribute is used only once in a document

Example

This example specifies the language of an XHTML Strict document as English:

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
Core Events

The last major aspect of modern markup initially introduced by HTML 4 was the increased possibility of adding scripting to HTML documents. In preparation for a more dynamic Web, a set of core events has been associated with nearly every HTML element. Most of these events are associated with a user doing something. For example, the user clicking an object is associated with an `onclick` event attribute. So,

```html
<p onclick="alert('Ouch!');">
Press this paragraph
</p>
```

would associate a small bit of scripting code with the paragraph event, which would be triggered when the user clicks the paragraph. In reality, the event model is not fully supported by all browsers for all tags, so the previous example might not do much of anything.

Block-Level Elements

The `<address>` tag

The `<address>` tag is used to surround information, such as the signature of the person who created the page, or the address of the organization the page is about. For example,

```html
<address>
Demo Company, Inc.<br />
1122 Fake Street<br />
San Diego, CA 92109<br />
</address>
```

can be inserted toward the bottom of every page throughout a Web site.

The `<address>` tag should be considered logical, although its physical rendering is italicized text. The HTML specification treats `<address>` as an idiosyncratic block-level element. Like other block-level elements, it inserts a blank before and after the block. It can enclose many lines of text, formatting elements to change the font characteristics and even images. However, according to the specification, it isn't supposed to enclose other block-level elements, although browsers generally allow this.

Text-Level Elements
Subject: web engineering

Text-level elements in HTML come in two basic flavors: physical and logical. Physical elements, such as `<b>` for bold and `<i>` for italic, are used to specify how text should be rendered. Logical elements, such as `<strong>` and `<em>`, indicate what text is, but not necessarily how it should look. Although common renderings exist for logical text elements, the ambiguity of these elements and the limited knowledge of this type of document structuring have reduced their use. However, the acceptance of style sheets and the growing diversity of user agents mean using logical elements makes more sense than ever.

**Physical Character-Formatting Elements**

Sometimes you might want to use bold, italics, or other font attributes to set off certain text, such as computer code. HTML and XHTML support various elements that can be used to influence physical formatting. The elements have no meaning other than to make text render in a particular way.

The common physical elements are:

<table>
<thead>
<tr>
<th>Element</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;i&gt;</code> … <code>&lt;i&gt;</code></td>
<td>Italics</td>
</tr>
<tr>
<td><code>&lt;b&gt;</code> … <code>&lt;b&gt;</code></td>
<td>Bold</td>
</tr>
<tr>
<td><code>&lt;tt&gt;</code> … <code>&lt;tt&gt;</code></td>
<td>Teletype (monospaced)</td>
</tr>
<tr>
<td><code>&lt;u&gt;</code> … <code>&lt;u&gt;</code></td>
<td>Uncertline</td>
</tr>
<tr>
<td><code>&lt;s&gt;</code> … <code>&lt;s&gt;</code></td>
<td>Strikethrough</td>
</tr>
<tr>
<td><code>&lt;strike&gt;</code> … <code>&lt;strike&gt;</code></td>
<td>Strikethrough</td>
</tr>
<tr>
<td><code>&lt;sub&gt;</code> … <code>&lt;sub&gt;</code></td>
<td>Subscript</td>
</tr>
<tr>
<td><code>&lt;sup&gt;</code> … <code>&lt;sup&gt;</code></td>
<td>Superscript</td>
</tr>
<tr>
<td><code>&lt;big&gt;</code> … <code>&lt;big&gt;</code></td>
<td>Bigger font (one size bigger)</td>
</tr>
<tr>
<td><code>&lt;small&gt;</code> … <code>&lt;small&gt;</code></td>
<td>Smaller font (one size smaller)</td>
</tr>
</tbody>
</table>

The following example code shows the basic use of the physical text-formatting elements:
Logical Elements
Logical elements indicate the type of content that they enclose. The browser is relatively free to determine the presentation of that content, although there are expected renderings for these elements that are followed by nearly all browsers. Although this practice conforms to the design of HTML, there are issues about designer acceptance. Plain and simple, will a designer think <strong> or <b> ? As mentioned previously, HTML purists push for <strong> because a browser for the blind could read strong text properly. For the majority of people coding Web pages, however, HTML is used as a visual language,
Subject: web engineering

despite its design intentions. Even when logical elements are used, many developers assume their default rendering in browsers to be static. `<h1>` tags always make something large in their minds. Little encourages Web page authors to think in any other way. Consider that until recently, it was almost impossible to insert a logical tag using a WYSIWYG HTML editor.

Seasoned experts know the beauty and intentions behind logical elements, and with style sheets logical elements will continue to catch on and eventually become the dominant form of page design. Even at the time of this writing, a quick survey of large sites shows that logical text elements are relatively rare. However, to embrace the future and style sheets, HTML authors should strongly reexamine their use of these elements. Table illustrates the logical text-formatting elements supported by browsers.

<table>
<thead>
<tr>
<th>Element</th>
<th>Meaning</th>
<th>Common Rendering</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;abbr&gt;</code> ... <code>&lt;abbr&gt;</code></td>
<td>Abbreviation (for example, Inc)</td>
<td>Plain</td>
</tr>
<tr>
<td><code>&lt;acronym&gt;</code> ... <code>&lt;acronym&gt;</code></td>
<td>Acronym (for example, WWW)</td>
<td>Plain</td>
</tr>
<tr>
<td><code>&lt;cite&gt;</code> ... <code>&lt;cite&gt;</code></td>
<td>Citation</td>
<td>Italic</td>
</tr>
<tr>
<td><code>&lt;code&gt;</code> ... <code>&lt;code&gt;</code></td>
<td>Code listing</td>
<td>Fixed Width</td>
</tr>
<tr>
<td><code>&lt;dfn&gt;</code> ... <code>&lt;dfn&gt;</code></td>
<td>Definition</td>
<td>Italic</td>
</tr>
<tr>
<td><code>&lt;em&gt;</code> ... <code>&lt;em&gt;</code></td>
<td>Emphasis</td>
<td>Italic</td>
</tr>
<tr>
<td><code>&lt;kbd&gt;</code> ... <code>&lt;kbd&gt;</code></td>
<td>Keyboard</td>
<td>Fixed Width</td>
</tr>
<tr>
<td><code>&lt;samp&gt;</code> ... <code>&lt;samp&gt;</code></td>
<td>Sample text (example)</td>
<td>Fixed Width</td>
</tr>
<tr>
<td><code>&lt;strong&gt;</code> ... <code>&lt;strong&gt;</code></td>
<td>Strong emphasis</td>
<td>Bold</td>
</tr>
<tr>
<td><code>&lt;var&gt;</code> ... <code>&lt;var&gt;</code></td>
<td>Programming variable</td>
<td>Italic</td>
</tr>
</tbody>
</table>

The following example uses all of the logical elements in a test document:
Subject: web engineering

Figure shows the rendering of the logical elements under Internet Explorer:
Logical Text Elements

WWW is an acronym
WWW is an abbreviation
This is Emphasis
This is Strong
This is Citation
This is Code
This is Definition
This is Keyboard
This is Sample
This is Variable
Lecture 19

Topic: Linking Basics, Linking in HTML, Images and Anchors, Anchor Attributes

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

<a href="url">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:
<a href="http://www.google.com">Visit google</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:
<a href="http://www.google.com/" target="_blank">Visit google</a>

Images and Anchors
Subject: web engineering

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 </a> tags on each side of the image. Below is the HTML code used to make the image work as a link to a page:

```html
<a href="xxx.htm"><img src="xxx.gif"></a>
```

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:

```html
<a href="xxx.htm"><img src="xxx.gif" border="0"></a>
```

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:

```html
<a href="xxx.htm"><img src="xxx.gif" border="0" alt="Link to this page"></a>
```

**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
Lecture 20

Topic: Image maps, Semantic Linking Meta Information

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.
Subject: web engineering

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

```html
<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap" />

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun" />
  <area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury" />
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus" />
</map>
```

**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 dom

*(meta tag and RDF covered in lecture 10)*
Subject: web engineering

Lecture 21

Topic: Image Preliminaries, Image Download Issues, Image as Buttons

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

```html
<img src="url" />
```

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

**HTML The `<img>` Tag and the Src Attribute**

In HTML, images are defined with the `<img>` tag. The `<img>` tag is empty, which means that it contains attributes only, and 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.

Syntax for defining an image:

```html
<img src="url" alt="some_text"/>
```

The URL points to the location where the image is stored. An image named "boat.gif", located in the "images" directory on "www.googlecom" has the URL: http://www.google.com/images/boat.gif.

The browser displays the image where the `<img>` tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

**HTML The Alt Attribute**
Subject: web engineering

The required alt attribute specifies an alternate text for an image, if the image cannot be displayed.
The value of the alt attribute is an author-defined text:
<img src="boat.gif" alt="Big Boat" />
The alt attribute provides alternative information for an image if a user for some reason cannot view it

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.

A button is marked up as follows:
<button type="button">Click Me!</button>

Definition and Usage

The <button> tag defines a push button.

Inside a button element you can put content, like text or images. This is the difference between this element and buttons created with the input element.

Always specify the type attribute for the button. The default type for Internet Explorer is "button", while in other browsers (and in the W3C specification) it is "submit".
Lecture 22

Topic: Introduction to Layout: Backgrounds, Colors and Text, Fonts, Layout with Tables

HTML 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.

<table>
<thead>
<tr>
<th>HTML</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;p&gt;text&lt;/p&gt;</td>
<td>Adds a paragraph break after the text.</td>
</tr>
<tr>
<td>&lt;p align=&quot;left&quot;&gt;text&lt;/p&gt;</td>
<td>Left justify text in paragraph.</td>
</tr>
<tr>
<td>&lt;p align=&quot;center&quot;&gt;text&lt;/p&gt;</td>
<td>Center text in paragraph.</td>
</tr>
<tr>
<td>&lt;p align=&quot;right&quot;&gt;text&lt;/p&gt;</td>
<td>Right justify text in paragraph.</td>
</tr>
<tr>
<td>&lt;br/&gt;</td>
<td>Adds a single linebreak where the tag is.</td>
</tr>
</tbody>
</table>
Subject: web engineering

Background color, Text and Fonts

Background color of web page, the text style and fonts can be selected differently.

Syntax (for background color)

<th bgcolor="value">

Attribute Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color_name</td>
<td>Specifies the background color with a color name (like &quot;red&quot;)</td>
</tr>
<tr>
<td>hex_number</td>
<td>Specifies the background color with a hex code (like &quot;#ff0000&quot;)</td>
</tr>
<tr>
<td>rgb_number</td>
<td>Specifies the background color with an rgb code (like &quot;rgb(255,0,0)&quot;)</td>
</tr>
</tbody>
</table>

Font:

Specify the font size, font face and color of text:

For example:

<font size="3" color="red">This is some text!</font>
<font size="2" color="blue">This is some text!</font>
<font face="verdana" color="green">This is some text!</font>

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...
Subject: web engineering

<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

```
- Header
- Left menu
  - Item 1
  - Item 2
  - Item 3...
- Main body
- Footer
```
Subject: web engineering

Lecture 23

Topic: Advanced Layout: Frames and Layers, HTML

HTML Frames

With frames, you 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 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 frameset element holds one or more frame elements. Each frame element can hold a separate document. The frameset element states HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

The Frame Tag

The `<frame>` tag defines what HTML document to put into each frame. In the example below we 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:

```html
<frameset cols="25%,75%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
</frameset>
```

Note: The frameset column size can also be set in pixels (cols="200,500"), and one of the columns can be set to use the remaining space, with an asterisk (cols="25%,*").
Subject: web engineering

You cannot use the <body></body> tags together with the <frameset></frameset> tags! However, if you add a <noframes> tag containing some text for browsers that do not support frames, you will have to enclose the text in <body></body> tags! See how it is done in the first example below.

### HTML Frame Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;frameset&gt;</td>
<td>Defines a set of frames</td>
</tr>
<tr>
<td>&lt;frame /&gt;</td>
<td>Defines a sub window (a frame)</td>
</tr>
<tr>
<td>&lt;noframes&gt;</td>
<td>Defines a noframe section for browsers that do not handle frames</td>
</tr>
<tr>
<td>&lt;iframe&gt;</td>
<td>Defines an inline sub window (frame)</td>
</tr>
</tbody>
</table>

#### <frame /> tag

**Definition and Usage**
The <frame> tag defines one particular window (frame) within a frameset. Each frame in a frameset can have different attributes, such as border, scrolling, the ability to resize, etc.

```html
<html>
  <frameset cols="25%,50%,25%">
    <frame src="frame_a.htm" />
    <frame src="frame_b.htm" />
    <frame src="frame_c.htm" />
  </frameset>
</html>
```

#### <noframes> tag:

**Example**

An inline frame is marked up as follows:

```html
<iframe src ="html_intro.asp" width="100%" height="300">
<p>Your browser does not support iframes.</p>
</iframe>
```

**Definition and Usage**
Subject: web engineering

The `<iframe>` tag defines an inline frame that contains another document.

```html
<noframes>
  <frame src="frame_c.htm" />
</noframes>
</noframes>
</frameset>
</html>
```

**Definition and Usage**

The `<noframes>` tag is used for browsers that do not handle frames. The noframes element can contain all the elements that you can find inside the body element of a normal HTML page. The noframes element is most used to link to a non-frameset version of the web site or to display a message to users that frames are required. The noframes element goes inside the frameset element.
Lecture 24

Topic: Other media types. Audio Support in Browsers, Video Support

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.

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
Subject: web engineering

The MIDI (Musical Instrument Digital Interface) is a format for sending music information between electronic music devices like synthesizers and PC sound cards. 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.. 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. 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.

**HTML Audio Tag:**

Example:

```html
<audio src="horse.ogg" controls="controls">
    <source src="horse.ogg" type="audio/ogg">
    Your browser does not support the audio element.
</audio>
```

**Definition and Usage**

The `<audio>` tag defines sound, such as music or other audio streams. The `<audio>` tag is new in HTML5. You can write text inside the start and end audio tags, to show older browser that they do not support this tag.

**Attributes**

New in HTML5.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoplay</td>
<td>autoplay</td>
<td>If present, the audio will start playing as soon as it is ready.</td>
</tr>
<tr>
<td>controls</td>
<td>controls</td>
<td>If present, controls will be displayed, such as a play button.</td>
</tr>
<tr>
<td>loop</td>
<td>loop</td>
<td>If present, the audio will start over again, every time it is finished.</td>
</tr>
<tr>
<td>preload</td>
<td>auto</td>
<td>Specifies if the audio should be loaded when the page loads. Ignored if autoplay is present.</td>
</tr>
<tr>
<td></td>
<td>metadata</td>
<td></td>
</tr>
<tr>
<td></td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>src</td>
<td>url</td>
<td></td>
</tr>
</tbody>
</table>

**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
Subject: web engineering

of Netscape and Internet Explorer. Videos stored in the Shockwave format have the extension .swf.

**HTML Video Tags:**

**Definition and Usage**

The `<video>` tag defines video, such as a movie clip or other video streams. The `<video>` tag is new in HTML5.

```html
<video src="movie.ogg" controls="controls">
  your browser does not support the video tag
</video>
```

You can write text between the start and end tags, to show older browser that they do not support this tag.

**Optional Attributes**

New in HTML5.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio</td>
<td>muted</td>
<td>Defining the default state of the the audio. Currently, only &quot;muted&quot; is allowed</td>
</tr>
<tr>
<td>autoplay</td>
<td>autoplay</td>
<td>If present, then the video will start playing as soon as it is ready</td>
</tr>
<tr>
<td>controls</td>
<td>controls</td>
<td>If present, controls will be displayed, such as a play button</td>
</tr>
<tr>
<td>height</td>
<td>pixels</td>
<td>Sets the height of the video player</td>
</tr>
<tr>
<td>loop</td>
<td>loop</td>
<td>If present, the video will start over again, every time it is finished</td>
</tr>
<tr>
<td>poster</td>
<td>url</td>
<td>Specifies the URL of an image representing the video</td>
</tr>
<tr>
<td>preload</td>
<td>preload</td>
<td>If present, the video will be loaded at page load, and ready to run. Ignored if &quot;autoplay&quot; is present</td>
</tr>
<tr>
<td>src</td>
<td>url</td>
<td>The URL of the video to play</td>
</tr>
<tr>
<td>width</td>
<td>pixels</td>
<td>Sets the width of the video player</td>
</tr>
</tbody>
</table>
Lecture 25

Topic: Other binary Formats. Style Sheets, Positioning with Style sheets

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

Style Sheets

CSS stands for Cascading Style Sheets. It is a way to divide the content from the layout on web pages.

How it works

A style is a definition of fonts, colors, etc. Each style has a unique name: a selector. The selectors and their styles are defined in one place. In your HTML contents you simply refer to the selectors whenever you want to activate a certain style.

Advantages

With CSS, you will be able to:

1) Define the look of your pages in one place rather than repeating yourself over and over again throughout your site.
2) Easily change the look of your pages even after they're created. Since the styles are defined in one place you can change the look of the entire site at once.
Subject: web engineering

3) Define font sizes and similar attributes with the same accuracy as you have with a word processor - not being limited to just the seven different font sizes defined in HTML.
4) Position the content of your pages with pixel precision.
5) Redefine entire HTML tags. Say for example, if you wanted the bold tag to be red using a special font - this can be done easily with CSS.
6) Define customized styles for links - such as getting rid of the underline.
7) Define layers that can be positioned on top of each other (often used for menus that pop up).

CSS Example

CSS declarations always ends with a semicolon, and declaration groups are surrounded by curly brackets:

```css
tp {color:red;text-align:center;}
```

To make the CSS more readable, you can put one declaration on each line, like this:

Example:

```css
text-align:center; }
```

**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

```css
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**
Subject: web engineering

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); nonfloated content will flow along the side opposite the specified float direction.
Lecture 26

Topic: Basic Interactivity and HTML Forms

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
Subject: web engineering

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 */

**HTML Forms**
Subject: web engineering

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

HTML Forms are used to select different kinds of user input. HTML forms are used to pass data to a server.

A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements. The <form> tag is used to create an HTML form:

<form>
input elements
</form>
Subject: web engineering

HTML Forms - The Input Element

The most important form element is the input element. The input element is used to select user information. An input element can vary in many ways, depending on the type attribute. An input element can be of type text field, checkbox, password, radio button, submit button, and more.

For example:

Text Fields
<input type="text" /> defines a one-line input field that a user can enter text into:
<form>
First name: <input type="text" name="firstname" /><br />
Last name: <input type="text" name="lastname" />
</form>

How the HTML code above looks in a browser:

Top of Form
First name:
Last name:
Subject: web engineering

Lecture 27

Topic: Form Control, New and emerging Form elements

Forms 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.

```html
<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.

```html
<form>
  <input type="radio" name="sex" value="male" /> Male
  <br />
</form>
```
Subject: web engineering

<input type="radio" name="sex" value="female" /> Female
</form>
Subject: web engineering

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.

```html
<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:

**HTML5 New Form Elements**

HTML5 has several new elements and attributes for forms.

- datalist
- keygen
- output

**Datalist Element**

The datalist element specifies a list of options for an input field.
The list is created with option elements inside the datalist.
Subject: web engineering

To bind a datalist to an input field, let the list attribute of the input field refer to the id of the datalist:

Example:

Webpage:

```html
<input type="url" list="url_list" name="link" />
<datalist id="url_list">
<option label="W3Schools" value="http://www.w3schools.com" />
<option label="Google" value="http://www.google.com" />
<option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

Tip: The option elements should always have a value attribute.

**Keygen Element**

The purpose of the keygen element is to provide a secure way to authenticate users. The keygen element is a key-pair generator. When a form is submitted, two keys are generated, one private and one public. The private key is stored on the client, and the public key is sent to the server. The public key could be used to generate a client certificate to authenticate the user in the future. Currently, the browser support for this element is not good enough to be a useful security standard.

Example:

```html
<form action="demo_form.asp" method="get">
Username: <input type="text" name="usr_name" />
Encryption: <keygen name="security" />
<input type="submit" />
</form>
```

**Output Element**

The output element is used for different types of output, like calculations or script output:

```html
<output id="result" onforminput="resCalc()"></output>
```
Subject: web engineering

Lecture 28

Topic: Java Server Pages and Active Server Pages: Basics

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
Subject: web engineering

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 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:

```jsp
<%@ page import="java.util.Date" %>
```
where to send the surfer in the event of a runtime Java problem:

```jsp
<%@ page errorPage="errorPage.jsp" %>
```
and whether you need to manage information at the session level for the user, possibly across multiple Web pages:

```jsp
<%@ 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:

```jsp
<%@ 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:
Subject: web engineering

```jsp
<%= 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:

```jsp
<% 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:

```jsp
<%-- comment for server side only --%>
```
Subject: web engineering

Lecture 29

Topic: Integrating Script, JSP Objects and Components

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 ) {
  <%
  String username = request.getParameter("username");
  if ( username != null && username.length() > 0 ) {
    <%
  }
%
}
```

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.
Subject: web engineering

The general syntax of JSP expression is as follows:

```jsp
<%= 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:

```jsp
<%= 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**
Subject: web engineering

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
Subject: web engineering

Lecture 30

Topic: JSP objects and components

The most commonly used implicit objects are request, response and session objects.

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:


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()}
Subject: web engineering

getLast AccessedTime
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. 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
Subject: web engineering

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 or false.

General syntax of isNew() of session object is as follows:

    session.isNew()

The returned value from the above method isNew() is Boolean
Lecture 31

Topic: ASP Objects and Components

ASP Objects

**ASP Response**: The ASP Response object is used to send output to the user from the server.

*Example:*

```html
<html>
<body>

<!--
response.write("Hello World!")
-->

</body>
</html>
```

*Output*

Hello World!

**ASP Request**

The Request object is used to get information from a visitor.

```html
<html>
<body>

<a href="demo_simplequerystring.asp?color=green">Example</a>

<!--
Response.Write(Request.QueryString)
-->

</body>
</html>
```

*Output*

Example

**ASP Application:**
A group of ASP files that work together to perform some purpose is called an application. The Application object is used to tie these files together.

**Application Object**

An application on the Web may consists of several ASP files that work together to perform some purpose. The Application object is used to tie these files together. The Application object is used to store and access variables from any page, just like the Session object. The difference is that ALL users share ONE Application object (with Sessions there is ONE Session object for EACH user).

The Application object holds information that will be used by many pages in the application (like database connection information). The information can be accessed from any page. The information can also be changed in one place, and the changes will automatically be reflected on all pages.

**ASP Session:**

A Session object stores information about, or change settings for a user session.

```html
<html>
<body>

<%
response.write("<p>")
response.write("Default Timeout is: " & Session.Timeout & " minutes.")
response.write("</p>")
Response.Write("<p>")
Session.Timeout=30
Response.Write("<p>")
Response.Write("Timeout is now: " & Session.Timeout & " minutes.")
Response.Write("</p>")
%

</body>
</html>
```

Output: Default Timeout is: 20 minutes.
Timeout is now: 30 minutes.

**ASP Server:**

The Server object is used to access properties and methods on the server.

```html
<html>
<body>
```
Subject: web engineering


<p><a href="text/textfile.txt"><img border="0" src="/images/btn_view_text.gif"></a></p>

Output:
Hello World line 1
Hello World line 2
Hello World line 3

ASP Error:
The ASPError object displays information about errors in scripts.

ASP FileSystem:
The FileSystemObject object is used to access the file system on a server.

<% Set fs=server.CreateObject("Scripting.FileSystemObject") If (fs.FileExists("c:\winnt\cursors\3dgarro.cur"))=true Then Response.Write("File c:\winnt\cursors\3dgarro.cur exists.") Else Response.Write("File c:\winnt\cursors\3dgarro.cur does not exist.") End If set fs=nothing %>
Subject: web engineering

</body>
</html>
Output: File c:\winnt\currors\3dgarro.cur exists.
Subject: web engineering

**ASP TextStream:**

The TextStream object is used to access the contents of a text file.

```html
<%= Set fs=Server.CreateObject("Scripting.FileSystemObject")
Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1) Response.Write(f.ReadAll) f.Close
Set f=Nothing
Set fs=Nothing
%>
</body>
</html>
```

**Output:**

This is the text in the text file:
Hello! How are you today?

**ASP Drive:**

The Drive object is used to get information about a local disk drive or a network share.

```html
<%= Dim fs,d,n
Set fs=Server.CreateObject("Scripting.FileSystemObject")
Set d=fs.GetDrive("c:")
   n = "Drive: " & d
   n = n & "<br />Total size in bytes: " & d.TotalSize
Response.Write(n)
set d=Nothing
set fs=Nothing
%>
</body>
</html>
```
Output:
Drive: C:
Total size in bytes: 4293563392

ASP File: The File object is used to return information about a specified file.

ASP Folder: The Folder Object is used to return information about a specified folder.

ASP Dictionary: The Dictionary object stores information in name/value pairs.

ASP ADO: ADO can be used to access databases from your web pages.

Accessing a Database from an ASP Page
The common way to access a database from inside an ASP page is to:

- Create an ADO connection to a database
- Open the database connection
- Create an ADO recordset
- Open the recordset
- Extract the data you need from the recordset
- Close the recordset
- Close the connection

What is ADO?
ADO is a Microsoft technology
ADO stands for ActiveX Data Objects
ADO is a Microsoft Active-X component
ADO is automatically installed with Microsoft IIS
ADO is a programming interface to access data in a database
Lecture 32

Topic: configuring and troubleshooting

ASP Components

ASP AdRotator: ASP AdRotator Component

The ASP AdRotator component creates an AdRotator object that displays a different image each time a user enters or refreshes a page. A text file includes information about the images.

Syntax
<% set adrotator=server.createobject("MSWC.AdRotator") adrotator.GetAdvertisement("textfile.txt") %>

ASP BrowserCap: ASP Browser Capabilities Component

The ASP Browser Capabilities component creates a BrowserType object that determines the type, capabilities and version number of a visitor's browser.

When a browser connects to a server, a User Agent header is also sent to the server. This header contains information about the browser.

The BrowserType object compares the information in the header with information in a file on the server called "Browscap.ini".

If there is a match between the browser type and version number in the header and the information in the "Browscap.ini" file, the BrowserType object can be used to list the properties of the matching browser. If there is no match for the browser type and version number in the Browscap.ini file, it will set every property to "UNKNOWN".

Syntax
<% Set MyBrow=Server.CreateObject("MSWC.BrowserType") %>
Subject: web engineering

**ASP Content Linking**: ASP Content Linking Component

The ASP Content Linking component is used to create a quick and easy navigation system!

The Content Linking component returns a Nextlink object that is used to hold a list of Web pages to be navigated.

**Syntax**

```asp
<% 
Set nl=Server.CreateObject("MSWC.NextLink") 
%>
```

**ASP Content Rotator**: ASP Content Rotator Component

The ASP Content Rotator component creates a ContentRotator object that displays a different content string each time a visitor enters or refreshes a page.

A text file, called the Content Schedule File, includes the information about the content strings.

The content strings can contain HTML tags so you can display any type of content that HTML can represent: text, images, colors, or hyperlinks.

**Syntax**

```asp
<% 
Set cr=Server.CreateObject("MSWC.ContentRotator") 
%>
```

**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.
Subject: web engineering

We suggest using, dreamweaver from adobe, or eclipse with myEclipse plugin, NetBeans from sun. Or sun studio creator from sun.

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).
Lecture 33

Topic: Request and response objects

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 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)
Subject: web engineering

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.
Subject: web engineering

`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:

```java
response.sendError(int status_code)
```
Subject: web engineering

Lecture 34

Topic: Request and response objects, Retrieving the contents of a an HTML form

SP 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:

- getAttributes()
- getHeader(String name)
- getHeaderNames()
- getAttribute(String name)
- getAttributeNames()
- getMethod()
- getParameter(String name)
- getParameterNames()
- getParameterValues(String name)
- getQuerySrting()
- 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")
Subject: web engineering

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.

General 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:
Subject: web engineering

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)

gParameterValue() 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
Subject: web engineering

returned as an 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()`
Subject: web engineering

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", "workonline");
```

The above example assigns the value “workonline” to username.

**removeAttribute(String)**
Subject: web engineering

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:

```java
request.removeAttribute(String);
```
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

```html
<form action="form.jsp" method="get">
<table>
<tr><td><b>Name</b></td><td><input type="text" name="name"></td></tr>
<tr><td><b>Favorite color</b></td><td><input type="text" name="color"></td></tr>
</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

```java
Name: <%= request.getParameter("name") %>
Color: <%= request.getParameter("color") %>
```
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
<html>
<head></head>
<body>

<jsp:include page="callee.jsp"/>
<jsp:param name="param2" value="value2"/>
<jsp:param name="param3" value="value3"/>
</jsp:include>

Callable:
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:
Subject: web engineering

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:

```java
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()`
Subject: web engineering

Returns the name of the cookie.
Subject: web engineering

**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.
Subject: web engineering

Lecture 36

Topic: Creating and Reading Cookies

Create a Cookie:

```html
<html>
<head>
<title>Reading a Cookie</title>
</head>

<body>
<h1>Reading a Cookie</h1>

```%
Cookie cookie1 = new Cookie("message", "Hello!");
cookie1.setMaxAge(24 * 60 * 60);
response.addCookie(cookie1);
%
<p>refresh to see the Cookie</p>
<% Cookie[] cookies = request.getCookies();

for(int i = 0; i < cookies.length; i++) {
    if (cookies[i].getName().equals("message")) {
        out.println("The cookie says " + cookies[i].getValue());
    }
}%

</body>
</html>
```

Read a Cookie:

```html
<html>
<head>
<title>Setting and Reading Cookies</title>
</head>

```
Subject: web engineering

```html
<BODY
<%
Cookie c = new Cookie("message", "Hello!");

    c.setMaxAge(24 * 60 * 60);
    response.addCookie(c);
    %>

<%
Cookie[] cookies = request.getCookies();
boolean foundCookie = false;

for(int i = 0; i < cookies.length; i++) {
    Cookie cookie1 = cookies[i];
    if (cookie1.getName().equals("color")) {
        out.println("bgcolor = " + cookie1.getValue());
        foundCookie = true;
    }
}

if (!foundCookie) {
    Cookie cookie1 = new Cookie("color", "cyan" );
    cookie1.setMaxAge(24*60*60);
    response.addCookie(cookie1);
}
%>

<H1>Setting and Reading Cookies</H1>
This page will set its background color using a cookie after refreshing.
</BODY>
</HTML>
```
Subject: web engineering

Lecture 37

Topic: Using application Objects and Events.

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.HttpServletRequest`

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:
Subject: web engineering

application.getAttribute("workonline");

The above statement returns the object workonline

**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:

```java
application.getAttributeNames();
```

For example:

```java
Enumeration onlinemca;
workonline=application.getAttributeNames();
```

The above example returns the attribute names available within the current application as enumeration in workonline.

**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:

```java
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:

```java
application.setAttribute("exvar", "workonline");
```

In the above example, the object exvar is stored with the object name workonline 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:

```java
application.removeAttribute(String objName);
```

For example:
Subject: web engineering

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()**
Subject: web engineering

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 workonline = application.getInitParameter("eURL");
In the above, the value of initialization parameter eURL is retrieved and stored in string workonline.
```

**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.

example:
```
Enumeration e;
e=application.getInitParameterNames();
```

**getResourceAsStream(Path)**
Subject: web engineering

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("/workonline.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);
Subject: web engineering

Lecture 38

Topic: Overview of advance features of XML

XML

XML stands for EXtensible Markup Language. It is a markup language much like HTML. It was designed to carry data, not to display data. Its tags are not predefined. You must define your own tags. XML is designed to be self-descriptive.

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

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.
Subject: web engineering

- Define your own tags
  - In XML, you define your own tags.

**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**
Subject: web engineering

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.

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.

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,
Subject: web engineering

or the information can be transformed and published to paper, and the data can be read by any XML aware browser or application.
Subject: web engineering

Lecture 39

Topic: overview of XML

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.

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.

XML Documents
The XML declaration
The first line of an XML document is the XML declaration. It's a special kind of tag: 
```xml
<?xml version="1.0"?>
```
The version 1.0 is the actual version of XML. The XML declaration makes clear that we're talking XML and also which version is used. The version identification will become important after new versions of XML are used.

The root element

All XML documents must have a root element. All other elements in the same document are children of this root element. The root element is the top level of the structure in an XML document.
Subject: web engineering

Structure of an XML page

```xml
<?xml version="1.0"?>
<root>
  <element>
    <sub-element>
      content
    </sub-element>
    <sub-element>
      content
    </sub-element>
  </element>
</root>
```

All elements must be nested. The level of nesting can be arbitrarily deep.

XML Attributes

Elements in XML can use attributes. The syntax is:

```xml
<element attribute-name = "attribute-value"> ... </element>
```

The value of an attribute needs to be quoted, even if it contains only numbers. Example:

```xml
<car color = "green">volvo</car>
```

The same information can also be defined without using attributes:

```xml
<car>
  <brand>volvo</brand>
  <color>green</color>
</car>
```

Avoid attributes

When possible try to avoid attributes. Data structures are more easy described in XML tags. Software that checks XML-documents can do a better job with tags than with attributes.

Converting XML to HTML for Display

There exist several ways to convert XML to HTML for display on the Web.

Using HTML alone
Subject: web engineering

If your XML file is of a simple tabular form only two levels deep then you can display XML files using HTML alone.
Subject: web engineering

**Using HTML + CSS**

This is a substantially more powerful way to transform XML to HTML than HTML alone, but lacks the full power and flexibility of the methods listed below.

**Using HTML with JavaScript**

Fully general XML files of any type and complexity can be processed and displayed using a combination of HTML and JavaScript. The advantages of this approach are that any possible transformation and display can be carried out because JavaScript is a fully general purpose programming language. The disadvantages are that it often requires large, complex, and very detailed programs using recursive functions (functions that call themselves repeatedly) which are very difficult for most people to grasp.

**Using XSL and Xpath**

XSL (eXtensible Stylesheet Language) is considered the best way to convert XML to HTML. The advantages are that the language is very compact, very sophisticated HTML can be displayed with relatively small programs, it is easy to re-purpose XML to serve a variety of purposes, it is non-procedural in that you generally specify only what you wish to accomplish as opposed to detailed instructions as to how to achieve it, and it greatly reduces or eliminates the need for recursive functions. The disadvantages are that it requires a very different mindset to use, and the language is still evolving so that many XSL processors in the Web servers are out of date and newer ones must sometimes be invoked through DOS.
Subject: web engineering

Lecture 40

Topic: Overview of XML

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 it 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.
Subject: web engineering

**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.