Chapter 6: HTML/XHTML I

References:

- Erik Wilde: World Wide Web — Technische Grundlagen. Springer, 1999, ISBN 3-540-64700-7, 641 Seiten.
- Eric Ladd, Jim O'Donnell, et al.: Using HTML 4, XML, and Java 1.2, Platinum Edition. QUE, 1999, ISBN 0-7897-1759-X, 1282 pages.
- Rainer Klute: Das World Wide Web. Addison-Wesley, 1996, ISBN: 389319763X.
- Dave Raggett, W3C: HTML 3.2 Reference Specification. [http://www.w3.org/TR/REC-html32.html]
- Dave Raggett, Arnaud Le Hors, Ian Jacobs (Eds.): HTML 4.01 Specification. W3C, Dec 24, 1999. [http://www.w3.org/TR/html4/]
- User's Guide to ISO/IEC 15445:2000 HyperText Markup Language (HTML) [http://www.cs.tcd.ie/15445/UG.html]
- XHTML [tm] 1.0: The Extensible HyperText Markup Language. W3C, Jan 26, 2000. [http://www.w3.org/TR/xhtml1]
- Stefan Münz: HTML-Dateien selbst erstellen — SELFHTML. [http://www.netzwelt.com/selfhtml/] [http://www.teamone.de/selfaktuell/]
- Ian Graham: Introduction to HTML. [http://www.utoronto.ca/webdocs/HTMLdocs/NewHTML/htmlindex.html]
- NCSA Beginner's Guide to HTML (no longer maintained). [http://www.ncsa.uiuc.edu/General/Internet/WWW/]

Objectives

After completing this chapter, you should be able to:

- develop web pages in strict HTML 4.0/XHTML 1.0.
- write syntactically correct HTML/XHTML.
- read the HTML and XHTML specifications.
- evaluate whether something is possible in HTML.
- write a short paragraph about differences between HTML versions.

The following constructs will not be treated in this course: Forms, support for Stylesheets and Javascript.

Overview

1. Introduction
2. Basic Structure, Head
3. Text, Fonts (Inline Elements)
4. Text Structure (Block Elements)

HTML: Introduction (1)

- HTML is the language of the web:
  - ◇ Web browsers (Netscape, Internet Explorer, etc.) display files written in the HTML syntax.
  - ◇ These files may contain also programs in Java-script or other languages.
  - ◇ They can also contain references to files in other formats, e.g. images in GIF, JPEG, PNG, TIFF.

    Also: Audio, Video, PDF, and and programs to be downloaded.
  - ◇ But the HTML file is the frame of the document and normally contains most of the text.

HTML: Introduction (2)

- HTML, the "Hypertext Markup Language", is a special syntax for text files.

  Not every sequence of characters is valid HTML. However, browsers do not print error messages, and always show some output.

- Browsers interpret HTML code and generate output on the screen or a printer (also speech output).

  Browsers are, e.g., Netscape, Mozilla, Internet Explorer, Opera, Lynx.

- For improving the portability, one should check the syntactical correctness with a "Validation Service".

  E.g. [`http://validator.w3.org/`]. The following link in the file itself permits easy checks: [`http://validator.w3.org/check/referer`].

HTML: Introduction (3)

- One can write HTML with normal ASCII text editors (with which one would also write C programs).

  E.g. emacs, vi, vim, nedit, pico, jove, Notepad, WordPad.

- However, there are also special WYSIWYG editors for HTML, which present the document like it will later be formatted in the browser.

  E.g.: Microsoft Frontpage Editor, Macromedia Dreamweaver, Allaire Homesite, HoTMetal, Viennasoft Internet Designer Pro, Claris Home Page. See also [`http://www.iwns.de/edit.html`].

- There are also conversion programs that generate HTML from other formats, e.g. Latex2html.

HTML: Introduction (4)

- HTML is an application of the Standard Generalized Markup Language (SGML).

- I.e. an HTML document must satisfy the general syntax rules of SGML, using the element types, attributes, and entities declared in the HTML DTD.

- HTML 4.01 defines 91 element types, of which 10 are contained only for compatibility reasons with earlier versions and should not be used anymore.

  These "deprecated" element types may be removed in future versions.

HTML Versions (1)

- At the end of 1990, a first Prototyp of Server and Client (auf NeXT-Workstations) was finished.

- In December 1991, the system is presented at a conference.

- In May 1992, an article by Tim Berners-Lee is published in the Proceedings of the Third Joint European Networking Conference.

  This article contains URLs, HTTP, HTML.

HTML Versions (2)

- The HTML version from May 1992 is still quite simple. It contains
  - ◇ Headlines of different levels
  - ◇ Lists (ordered, unordered, definition lists)
  - ◇ Hypertext-Links (with the element `A`).
  - ◇ Title, Address, IsIndex [?]
  - ◇ Preformatted Text (for Listings) [?]
- Dave Raggett designed at the end of 1991 an extended version (HTML+). It was implemented in the browser "Arena".

HTML Versions (3)

- 1993 the browser Mosaic was published (Marc Andreesen and Eric Bina, NCSA). It also had HTML extensions.
- In November 1995, HTML 2.0 was published as Internet Proposed Standard RFC 1866.

  In July 1994 a HTML 2.0 draft was presented at an IETF meeting, and a HTML working group was formed. HTML 2.0 is based on the current practice in 1994.
- HTML 2.0 was the first version that was formally defined by an SGML DTD.

HTML Versions (4)

- At the end of 1994, Netscape was founded, and their browser again contained new element types.
- At the end of 1994 also the World Wide Web Consortium (W3C) was founded. It starts to work on HTML 3.0.
- HTML 3.0 was published as Internet Draft, but never formally adopted as a standard, since during the ratification process it became clear that it was already outdated.

HTML Versions (5)

- The next official HTML version after HTML 2.0 is HTML 3.2 (released in January 1997).

  It aims to capture the recommended practice in early 1996.
- Already in December 1997, HTML 4.0 is published as an official W3C Recommendation, editorial changes are done in April 1998.
- Important new features are: Style sheets, frames, improved tables, improved forms, general inclusion of multimedia objects.

HTML Versions (6)

- Three DTDs were published for HTML 4.0:
  - ◇ Strict HTML 4.0 contains fewer presentation oriented element types and attributes.

    One should instead use stylesheets.
  - ◇ Transitional HTML 4.0 is compatible with earlier versions and can still be used for some time.
  - ◇ The HTML 4.0 Frameset DTD is used for documents that describe how the browser window is divided into frames.

    The frame contents is written in Strict/Transitional HTML 4.0.

HTML Versions (7)

- For best compatibility with future HTML versions, it is recommended to use only the strict DTD.

  E.g. XHTML 1.1 (see below) corresponds only to strict HTML 4, the presentation oriented elements were left out. However, because XHTML is modularized, it would be possible to define modules that extend it corresponding to the transitional DTD.
- However, when HTML 4 was defined, not all browsers supported stylesheets yet.

  One can define stylesheets in a way that such browsers ignore them, but the web pages then are then displayed quite "spartanic". The transitional DTD gave a way to define presentation-oriented information in a way the browsers understood. Plus, existing HTML documents with presentation-oriented elements could still qualify as HTML 4.

HTML Versions (8)

- HTML 4.01 was published December 24, 1999 and corrects some errors in the earlier specification.

  [http://www.w3.org/TR/html401/]
- Besides the W3C Recommendation, there is also an ISO Standard (ISO/IEC 15445:2000) for HTML, which contains a subset of the W3C HTML 4.

  [http://www.cs.tcd.ie/15445/15445.html]
- HTML 4.01 is probably the last version that is based on SGML. Future versions are based on XML.

HTML Versions (9)

- XHTML 1.0 is a reformulation of HTML 4 in XML (W3C Recommendation January 26, 2000).

  [http://www.w3.org/TR/xhtml1/]
  It can be used in a way that is compatible with most existing browsers. And of course, it is supported by XML software. "The XHTML family is the next step in the evolution of the Internet." [XHTML Rec.] XHTML stands for "Extensible HyperText Markup Language".
- Then work was done on separating the DTD into modules. The W3C Recommendation "Modularization of XHTML" was published April 10, 2001.

  [http://www.w3.org/TR/xhtml-modularization/]
  [http://www.w3.org/MarkUp/modularization]

HTML Versions (10)

- XHTML Basic is a collection of modules for core HTML. (W3C Rec., December 19, 2000.)

  [http://www.w3.org/TR/xhtml-basic/].
  The required modules are the structure module (body, head, html, title), the text module (abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var), the hypertext module (a), and the list module (dl, dt, dd, ol, ul, li). Several additional modules are defined.

- XHTML 1.1 ("Module Based XHTML") is a reformulation of XHTML 1.0 strict in the module framework. (W3C Recommendation, May 31, 2001).

  [http://www.w3.org/TR/xhtml11/]

HTML vs. XHTML (1)

- In HTML, element and attribute names are case-insensitive. In XHTML, they must be lowercase.
- In HTML, many tags are optional (see the element declarations below). In XHTML, all start- and end-tags are required.
- In HTML, for empty elements only the start tag is written. In XHTML, one must use an empty-element tag, e.g. "
".

  For compatibility with existing browsers, it is recommended to put a space before the end of the tag, e.g. "
".

HTML vs. XHTML (2)

- In HTML, attribute values do not require quotes if they are valid name tokens (e.g. names, numbers). In XHTML, attribute values must always be quoted.

  E.g. in HTML, one can write <td colspan=2> or <ul type=square>. In XHTML, quotes are needed, e.g. <td colspan="2">.

- In HTML, attribute minimization can be used, e.g. one can write "<dl compact>". In XHTML, always attribute name and value are required:

```
<dl compact="compact">
```

  Some browsers are confused by the long form.

HTML vs. XHTML (3)

- In XHTML, an XML declaration is needed if one wants to use a character encoding different from UTF-8 (or UTF-16):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

  This can confuse some browsers.

  It might be safer to use UTF-8 and not to write an XML-declaration. Actually, safest would be to use only the ASCII subset of UTF-8.

- In XHTML, an XML namespace declaration is required in the document element html:

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

HTML vs. XHTML (4)

- In HTML, elements `script` and `style` have content model `CDATA`. In XHTML, they have content model `#PCDATA`.

  This means that in XHTML, “`&`” is interpreted, and `<`” is forbidden. One can use a `CDATA` section “`<![CDATA[...]]>`” to avoid the interpretation of the script or stylesheet as HTML. Unfortunately, this may confuse browsers. One should use external stylesheets/scripts if it contains “`&`”, `<`”, “`]]>`”, or “`--`”. Hiding scripts or stylesheets in comments does not work with XML, since XML processors may remove comments.

HTML vs. XHTML (5)

- In XHTML, the attribute `id` should be used for giving elements (e.g. `a`) a name that can be referenced in fragment identifiers in URIs. In HTML, one usually uses the attribute “`name`”.

  In HTML 4, “`id`” is already permitted, but not all browsers understand such fragment identifiers. In HTML 3.2, “`id`” was not permitted (only “`name`”), but browsers ignore attributes they do not understand. It is safest to use both attributes to assign the same fragment identifier to an “`a`” element.

HTML Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type"
            content="text/html; charset=iso-8859-1">
        <title>My first HTML document</title>
    </head>
    <body>
        <h1>Greeting</h1>
        <p>Hello, world!
    </body>
</html>
```


XHTML Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC
    "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://ww.w3.org/1999/xhtml">
    <head>
        <title>My first XHTML document</title>
    </head>
    <body>
        <h1>Greeting</h1>
        <p>Hello, world!</p>
    </body>
</html>
```

Overview

1. Introduction
2. Basic Structure, Head
3. Text, Fonts (Inline Elements)
4. Text Structure (Block Elements)

Internationalization (1)

- Texts can contain names or quotes in other languages.
- Nearly all HTML elements have internationalization attributes which are declared in the entity “`i18n`”:

```
<!ENTITY % i18n
          "lang %LanguageCode; #IMPLIED
          dir (ltr|rtl) #IMPLIED">
```

- “`i18n`” is an abbreviation for “internationalization”.

  “i”, 18 letters, “n”.

Internationalization (2)

- Knowing the language of the text or text pieces is useful for
  - ◇ Search engines
  - ◇ Typesetting programs

    E.g. for ligatures, hyphenation.
  - ◇ Spelling checkers
  - ◇ Speech output (for blind people, drivers).

Internationalization (3)

- The attribute `lang` identifies the language of the text inside the element, e.g. “`en`” for English, “`de`” for German, “`fr`” for French.

  `ar`: Arabic, `es`: Spanish, `el`: Greek, `he`: Hebrew, `hi`: Hindi, `it`: Italian, `ja`: Japanese, `ko`: Korean, `nl`: Dutch, `pl`: Polish, `pt`: Portuguese, `ru`: Russian, `ro`: Romanian, `th`: Thai, `tr`: Turkish, `zh`: Chinese.

- The language can be specialized with a country code, e.g. `en-GB`, `en-US`, `fr-FR`, `fr-BE`, `fr-CA`.

  Language codes: ISO standard 639. Country codes: ISO 3166.
  [http://www.oasis-open.org/cover/iso639a.html]
  [http://www.oasis-open.org/cover/country3166.html]
  See also RFC 1766.

Internationalization (4)

- If the attribute "`lang`" is not defined for an element, the value of its next ancestor in the tree is inherited.
- E.g. if the document is in one language, it suffices to define "`lang`" in the "`html`" element at the root.

  If even there "`lang`" is not defined, the value of the HTTP-header "`Content-Language`" is used.
- In XHTML, one should use both, `xml:lang` and `lang`.
- The attribute `dir` defines the writing direction for text and tables (e.g. `ltr`: from left to right).

  However, the Unicode standard also defines the writing direction.

General Attributes (1)

- Nearly all elements also have these attributes:

```
<!ENTITY % coreattrs
          "id         ID            #IMPLIED
          class       CDATA         #IMPLIED
          style       %StyleSheet;  #IMPLIED
          title       %Text;        #IMPLIED">
```

  - ◇ `id` is a unique identifier for the element
  - ◇ `class` and `style` are used for stylesheets
  - ◇ `title` is displayed by some browsers as "tool tip" when the mouse pointer is over the element.

General Attributes (2)

- The entity "`%events`" contains attributes that permit to specify program code (e.g. in javascript) that is executed when an event occurs while the mouse pointer is over the element.

  `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`, `onkeypress`, `onkeydown`, `onkeyup`.
- The entity `%attrs;` contains these three kinds of general attributes:

```
<!ENTITY % attrs "%coreattrs; %i18n; %events;">
```


The HTML Element (1)

- Every HTML document has an element of type `html` as document element (root of the element tree).
- I.e. HTML documents are always enclosed in

```
<html> ... </html>
```

  However, both tags are optional.

  There should also be a `DOCTYPE` declaration plus possibly an XML declaration in front of it.
- The element type `html` is declared as follows:

```
<!ELEMENT html O O (head, body)>
```

The HTML Element (2)

- `html` has only the internationalization attributes:

```
<!ATTLIST html %i18n;>
```

- It is good style to define the language of the document in the `html` element, e.g.

```
<html lang="de">
```


The Document Head (1)

- Each HTML document contains a "`head`"-element with information about the document, like
  - ◇ title,
  - ◇ meta data for search engines,
  - ◇ typed links to other documents,
  - ◇ style sheets,
  - ◇ program code (e.g. in JavaScript).
- The data in the `head` element does not appear in the main document window.

  E.g. the title is printed in the window headline.

The Document Head (2)

- The element type "`head`" is declared as follows:

```
<!ELEMENT head O O (title & base?)
                   +(script|style|meta|link|object)>
```

- I.e. within the `head` element, a `title` element is required, and a `base` element is permitted.

  Both elements can appear in arbitrary sequence, but each only once.

- In addition, the `head` element can contain elements of the types `script`, `style`, `meta`, `link`, and `object`.

  Since they are introduced as exception (inclusion), they can appear in arbitrary sequence before, after, and between `title` and `base`, and they can appear any number of times.

The Document Head (3)

- The element type `head` has only the internationalization attributes and an attribute `profile`, which refers to meta data (see below):

```
<!ATTLIST head %i18n; -- lang, dir --
               profile %URI; #IMPLIED>
```

- XHTML is completely compatible, but the content model is more complicated in XML:

```
<!ENTITY  % head.misc "(script|style|meta|link|object)*">
<!ELEMENT head (%head.misc;,
         ((title, %head.misc;, (base, %head.misc;)?) |
          (base, %head.misc;, (title, %head.misc;))))>
```

The Document Title (1)

- The `head` element must contain exactly one `title` element.
- The element type `title` is declared as follows:

```
<!ELEMENT title - - (#PCDATA)
                -(script|style|meta|link|object)>
<!ATTLIST title %i18n;>
```

- Within the `title` only pure text is permitted.

  The inclusions under “`head`” must be explicitly excluded here. The definition in the XHTML DTD is simpler since it did not (and cannot) use inclusions for the `head` element: `<!ELEMENT title (#PCDATA)>`

The Document Title (2)

- The title text should not be too long in order to fit into the window headline (e.g. not more than 64 characters).
- Search engines normally show the document title in their result. It should be understandable without context.

  Some search engines give documents containing the search terms in the title higher weight in the ranking of the search results. In AltaVista, the first 8 words in the title have especially high weight.

Meta Data (1)

- “`meta`” elements contain meta data about the document, especially information for search engines.

  Meta data are “data about data”. E.g. in a relational database, there are system tables that contain e.g. the names of all tables (plus the creation date etc.). These tables contain meta data.

- E.g. one can define keywords under which the document is entered into the index of search engines:

```
<meta name="keywords"
      content="html, halle, course" />
```

  The commas are not important for most search engines. Some search engines give explicitly defined keywords higher weight over words that simply appear in the text. Some ignore the the `meta` information.

Meta Data (2)

- One should also define a short description of the document (abstract) which search engines show in their result list:

```
<meta name="description"
      content="Homepage of the web course held
               in the summer term 2003 at the
               University of Halle.">
```

- If one does not define a description, many search engines show only the first few lines, which are often not very helpful.

Meta Data (3)

- One can also tell search engines that this page should not be entered into the index, and that it should not follow links in this page:

```
<meta name="ROBOTS"
      content="NOINDEX, NOFOLLOW">
```

- But many web robots (which collect data for search engines) look only for the file “robots.txt” in the root directory of the web server:

```
User-agent: *     # Applies to all robots
Disallow: /local  # Prefix of disallowed URIs
```


Meta Data (4)

- Meta tags can also be used to label pages with content that might be harmful to young persons.
- Browsers can be configured in such a way that they do not display pages with certain such labels.

PICS (Platform for Internet Content) [http://www.w3.org/PICS/] was designed for this purpose, but can contain also other information. PICS itself does not contain a rating system, and there are several different rating services that can be used. Internet Explorer uses RSACi ratings (“Recreational Software Advisory Council”). This is quite old and was replaced by the ICRA system (“Internet Content Rating Association”) [http://www.rsac.org]. One can use different rating systems with IE see [http://www.rsac.org/_en/support/en_msie.html]. An IE definition file for ICRA is [http://www.rsac.org/_en/support/icra.rat].

Meta Data (5)

- An example for a bad label in the ICRA system is:

```
<meta http-equiv="PICS-Label" content='(PICS-1.1
      "http://www.icra.org/ratingsv02.html"
      labels generic false
      for "http://www.icra.org/upgrade/icra2.html"
      ratings (ca 1 cb 1 la 1 lb 1 lc 1 na 1 nb 1 ...)
      "http://www.rsac.org/ratingsv01.html" l gen false
      for "http://www.icra.org/upgrade/icra2.html"
      r (n 4 s 4 v 4 l 4))' />
```

This label contains two ratings. The new ICRA system (first part) uses boolean values in many different categries, e.g. “la 1” means “explicit sexual language”. The second part are the RSCAi ratings with the categories nudity, sex, violence, language (0 to 4). Note that “labels” can be abbreviated to “l”, “generic” to “gen”, “ratings” to “r.”.

Meta Data (6)

- One can also specify HTTP headers in the document itself:

```
<meta http-equiv="Expires"
      content="Thu, 01 Feb 2001 16:29:00 GMT" />
```

- Web servers can use this information, but they seldom do, since they would have to parse the document before they deliver it.
- However, when a browser discovers such a meta element in the document, it should behave in the same way as if it had got the HTTP header.

Meta Data (7)

- The W3C validator requires that the character encoding is defined. If necessary, use e.g.

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

- One often finds the following construct to load another page, e.g. after 5 seconds:

```
<meta http-equiv="Refresh"
      content="5; URL=http://www.x.com/" />
```

- But this is illegal (HTTP has no "`Refresh`" header).

  The HTTP server should send e.g. the status code "301: Moved Permanently" with the new URI in the "`Location:`" header.

Meta Data (8)

- The element type "`meta`" is declared as follows:

```
<!ELEMENT meta - O EMPTY>
<!ATTLIST meta %i18n;
               http-equiv NAME  #IMPLIED
               name       NAME  #IMPLIED
               content    CDATA #REQUIRED
               scheme     CDATA #IMPLIED>
```

- The attribute `content` is required, and exactly one of the attributes `name` or `http-equiv` should be used.
- The attribute `scheme` is intended for specifying the interpretation of the `content` (e.g. date format).

Meta Data (9)

- The HTML specification does not define a set of values for the attribute `name` (meta data properties).

  `name="Author"` is often used in examples.

- The attribute `profile` of the `head`-element can refer to a definition of meta data properties.

  The value of `profile` is a URI, under which probably a human readable description of the valid values for "`name`" is stored (and also for the attributes "`rel`" and "`ref`" of "`link`" and "`a`"). If one such profile (or a small set of them) will become generally accepted, the URI will be built into browsers and search engines as a unique identification of the profile, even though they cannot understand the document itself.

Meta Data (10)

- The "Dublin Core" is a set of meta data properties ("Element Set") that is widely used.

  [http://purl.org/dc] [http://dublincore.org/documents/dces/]
  [http://dublincore.org/documents/1998/09/dces/]
  [http://www.ukoln.ac.uk/metadata/dcdot/]
  [http://www.w3.org/TR/rdf-primer/]
  The Dublin Core set was originally developed at the March 1995 Metadata Workshop in Dublin, Ohio [RDF Primer].

- The "Dublin Core" properties are: `Title`, `Creator`, `Subject`, `Description`, `Publisher`, `Contributor`, `Date`, `Type`, `Format`, `Identifier`, `Source`, `Language`, `Relation`, `Coverage`, `Rights`.

Meta Data (11)

- Meta data are important for powerful search tools and the future semantic web.
- The W3C wants to encode meta data in the "Resource Description Framework" RDF.
- However, RDF itself is only a data model.
- It can use different vocabularies/ontologies for the meta data properties, but in examples, often the Dublin Core is used.

The Document Body (1)

- The text of the document (contents of the browser window) is contained in the element "body".
- HTML distinguishes two kinds of elements:
  - ◇ Inline elements can appear within normal text and do not require a line break:

    ```
    <!ENTITY % inline "#PCDATA | tt | i | ...">
    ```
  - ◇ Block-level elements describe larger structures. For them, the browser normally starts a new line:

    ```
    <!ENTITY % block "p | h1 | h2 | h3 | ...">
    ```


The Document Body (2)

- The contents of body is a sequence of block elements, e.g. paragraphs (p) and headlines (h1, . . .).
- Within the paragraphs etc. the real text is written (i.e. "inline" contents).
- The element type "body" is declared as follows:

```
<!ELEMENT body O O (%block;|script)+
                   +(ins|del)>
<!ATTLIST body %attrs;
               onload   %Script; #IMPLIED
               onunload %Script; #IMPLIED>
```


The Document Body (3)

- In HTML 3.2, one could write text directly within the body element, now text should be enclosed in paragraphs (p elements) etc.
- In XHTML, the body can be empty, while in HTML, it cannot.
- The elements ins and del can be used to mark revisions of a text (insertions and deletions).

  They are special because they can appear as block-level elements and as inline elements. If they are used as inline elements, they cannot contain block-level content. Internet Explorer marks inserted text underlined, and deleted text crossed out.

The Document Body (4)

- In HTML 3.2/HTML 4.01 transitional one can set various colors with attributes of the **body** element:
  - ◇ **bgcolor**: Background color (paper).

    One can use color names (aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow) or specify RGB-values in hexadecimal notation: #RRGGBB.
  - ◇ **text**: Text color.
  - ◇ **link**: Color for hyperlinks, target was not visited.
  - ◇ **vlink**: Color for hyperlinks, target was visted.
  - ◇ **alink**: Color for "active" link (mouse is over it).
  - ◇ **background**: URI of background picture/texture.

The Document Body (5)

- In "strict HTML 4.01", **body** has only the general attributes and two attributes for JavaScript etc.
- Now things like colors should be defined in stylesheets. E.g. one can put the following into the **head**:

```
<style type="text/css">
    body {
        color: fuchsia;
        background-color: aqua;
    }
</style>
```


Overview

1. Introduction
2. Basic Structure, Head
3. Text, Fonts (Inline Elements)
4. Text Structure (Block Elements)

Text, Entity References (1)

- Normal text can be directly entered.
- If the editor does not support national characters or one wants to be very portable, one can enter national characters with character or entity references:
  - ◇ **ä** for "ä" (a-Umlaut), **Ä** for "Ä",
  - ◇ **ö** for "ö" (o-Umlaut), **Ö** for "Ö",
  - ◇ **ü** for "ü" (u-Umlaut), **Ü** for "Ü",
  - ◇ **ß** for "ß" (s-z-Ligature).

Text, Entity References (2)

- Other accents are:
  - ◇ `À`: “Á” (Accent grave),
  - ◇ `&Aactute;`: “À” (Accent aigu),
  - ◇ `Â`: “Â” (Accent circonflexe)
  - ◇ `Ã`: “Ã” (Tilde),
  - ◇ `Å`: “Å” (Ring),
  - ◇ `ç`: “ç” (Cédille).
- Entities for other letters with accents are named in the same way, e.g. “`î`” is an “î”.

  But only selected combinations of letter and accent are supported.

Text, Entity References (3)

- Since the characters <, >, and & have a special meaning, they must be encoded with entities if they appear in normal text:
  - ◇ `<` for “<”
  - ◇ `>` for “>”
  - ◇ `&` for “&”

    Also in attribute values (e.g. URLs), one must write `&` as `&`.
  - ◇ `"` for “ `"` ”

    This must be used inside attribute values delimited with `"`.
    But one can delimit the attribute value also with “ ’ ”.

Text, Entity References (4)

- Some other special characters are:
  - ◇ `§` gives the paragraph-symbol (section) §
  - ◇ `©` gives the copyright-symbol ©
  - ◇ `®` gives the “Registered Trademark” symbol
  - ◇ `µ` gives a “μ”.
  - ◇ `€` gives an “Euro” symbol.
- One can also use character references in HTML, e.g. write “`ä`” for “ä”.

Text, Entity References (5)

- HTML 4.01 is based on the Unicode/ISO 10646 character set.

  This defines the repertoire of characters that can be used and the codes that the system internally uses (i.e. codes that one must use in character references).
- Not every browser might be able to display the entire set of Unicode characters.
- Independent from this, one can use different encodings when one stores HTML documents in files or exchanges them over the internet.

Text, Entity References (6)

- E.g. ISO Latin 1 (ISO-8859-1) permits to store national characters of many European languages without entity/character references.
- But e.g. in Russia, one would normally use an encoding that supports cyrillic letters (e.g. ISO-8859-5).
- It is only important that the browser knows what character set is used (it translates from a sequence of bytes to a sequence of characters).

Text, Entity References (7)

- Normally, the web server should specify the character set with the following HTTP header:

  `Content-Type: text/html; charset=ISO-8859-1`
- The HTTP specification states that ISO Latin 1 (i.e. `ISO-8859-1`) is the default.
- But the HTML specification states that experience has shown that this is not reliable, and the default should not be used.

Text, Entity References (8)

- Therefore, if the HTTP server does not specify the character set, one should specify it with a `meta` element inside the document (see slide 6-45).

  This of course works only if the used character set is a superset of ASCII, and up to this `meta` tag only ASCII characters are used.
- If one uses XHTML without XML declaration, one must use UTF-8.

  The national characters with the ISO Latin 1 encoding cannot be used in this case (they would be represented as two bytes in UTF-8), but ASCII characters are of course safe.

Text, Line Breaks (1)

- The browser normally determines line breaks itself, since the HTML author does not know
  - how wide the browser window is,
  - which font the user has selected.

    One can specify a font via stylesheets, but it is e.g. possible that the selected font is not available and substituted by another one.
- The browser normally puts words into the current line until the next word does not fit. Then it starts a new line. Thus, no horizontal scrollbar is necessary.

  As long as no very long words, tables, frames, `pre` etc. are used.

Text, Line Breaks (2)

- Line breaks in the HTML source are not relevant: An arbitrary sequence of spaces, `TAB`s, form feeds, and line breaks is treated like a single space.
- The browser normally breaks lines only at word boundaries (this may depend on the browser).
- Suggested hyphenation positions in words can be marked with the entity `` ("soft hyphen").

  This is character number 173 in the ISO Latin 1 code. If at this character a new line is started, it appears as "-" at the end of the previous line. Otherwise it is not displayed.

Text, Line Breaks (3)

- One can replace the space between two words by ` ` ("no-break space"). Then no line break is done between these two words.

  They are treated in the browser line a single word, i.e. ` ` (character 160 in ISO Latin 1) is a letter that looks like a space.

- Example:

```
E.g. after an abbre&shy;viation period,
a line break looks bad. Also use &amp;nbsp; in
7 kg, Exercise 3, S. Brass.
```

  If one uses ISO Latin 1, and the editor understands it, no entity references are required.

Sequences of Spaces

- In HTML, one can (nearly) always use a sequence of spaces, tabs, and line breaks instead of a single space.
- The browser only displays a single space, even if one writes five empty lines.
- If several spaces should appear in the output, one can use e.g. `   `.

  However, this is very presentation-oriented and not explicitly mentioned in the specification.

Line Breaks: `br`

- One can request line breaks with the element `br` ("forced line break"):

```
First line.<br />
Second line.
```

- The element `br` is declared as follows:

```
<!ELEMENT br - O EMPTY>
<!ATTLIST br %coreattrs;>
```

- `br` is an inline element, i.e. it is permitted whereever normal text can appear.

  In HTML 3.2, `br` had an attribute `clear` for moving below an image at the border. Again, one should now use stylesheets.

Preformatted text: pre (1)

- Text enclosed in "<pre>...</pre>" is displayed with the line breaks given in the HTML source.

  Otherwise one must replace all spaces by , and use
 at every line end to enforce that line breaks are done exactly as given.

- Within the "pre" element, also sequences of several spaces are respected, whereas normally, they are merged to a single space.

  The TAB-character (ASCII 9) is interpreted as the smallest number of spaces (at least one) that moves the cursor to the next tab stop, where tap stops are set every eight characters. However, the HTML specification strongly discourages using tabs and only says that browsers usually behave in this way.

Preformatted text: pre (2)

- Browsers usually show the contents of the "pre" element in teletype font ("fixed pitch"), so that also the given columns are respected.

  Program code is a typical application of "pre".

- Note that "pre" does not mean "verbatim": Markup in the contents of pre is interpreted.

  It seems that "style" and "script" are the only HTML elements with the content model "CDATA" of SGML. There is no "verbatim" element type. E.g. in order to show example HTML code, one must escape e.g. "<" as "<". In early HTML versions, there were LISTING, XMP, PLAINTEXT. But they were already in HTML 3.2 deprecated.

Preformatted text: pre (3)

- The element type pre is declared as follows:

```
<!ELEMENT pre - - (%inline;)*
              -(img|object|big|small|sub|sup)>
<!ATTLIST pre %attrs;>
```

- Changes of the font size and images are excluded inside "pre" in order not to disturb the grid of lines and columns.

- pre belongs to the block-level elements.

Linebreaks and Tags

- SGML requires that linebreaks directly after start tags and directly before end tags are ignored.

- Therefore

```
<pre>
First line.
Second   line.
</pre>
```

  is equivalent to:

```
<pre>First line.
Second   line.</pre>
```

- However, pre always automatically starts a new line.

Spaces and Tags

- Spaces after start tags and before end tags should not be ignored by a correct browser (only line breaks must be ignored at these places).
- But, at least earlier, some browsers eliminated all white space after start tags/before end tags.
- Therefore, instead of e.g.

```
Abc<tt> def </tt>ghi
```

  one should better write:

```
Abc <tt>def</tt> ghi
```


Markup of Words/Phrases (1)

- Words or phrases within the text can be marked, e.g. as program code, as defining occurrence of a term, as emphasized, etc.
- Normally, browsers display such words/phrases in another font (e.g. boldface or italics).
- Therefore, one can view these elements as logical font specifications.

  In contrast, physical font specifications explicitly select a specific font.

Markup of Words/Phrases (2)

- The classification of words/phrases can also be useful for generating an index or for advanced search engines.
- Example:

```
<dfn>HTTP</dfn> is the protocol used for the
communication between browsers and web servers.
<cite>Brass: Foundations of the WWW, 2003</cite>
```

- Depending on the browser, `dfn` (defining occurrence of a term) and `cite` (citation) may be both displayed as *italics*.

Markup of Words/Phrases (3)

- `em`: Emphasis (e.g. *italics*).
- `strong`: Strong emphasis (e.g. **boldface**).
- `cite`: Reference to another source (e.g. *italics*).
- `dfn`: Defining occurrence of a word (e.g. *italics*).
- `code`: Program code (e.g. `teletype`).
- `samp`: Example output of a program (e.g. `teletype`).
- `kbd`: Example input to a program (e.g. `teletype`).
- `var`: Variable or program argument (e.g. `teletype`).

Markup of Words/Phrases (4)

- HTML 4 has added two phrase elements:
  - ◇ `abbr`: Abbreviation (e.g. normal font)
  - ◇ `acronym`: Acronym (e.g. normal font)

    The difference between these elements is not clear. An acronym is an artificial word built from the first letters of several words. E.g. radar: "radio detecting and ranging". But in the HTML 4.01 specification, "WWW" is an `abbr` and "F.B.I." an `acronym`.
- `abbr/acronym` are e.g. important for: Speech output, spelling checkers, automatic translations.

  E.g. a speech generator could read the single letters (WWW), or search the abbreviation in a table for the full word (Dr. → doctor). The attribute `title` can be used for specifying the full text.

Markup of Words/Phrases (5)

- `cite` can be useful for advanced search engines: Where does "Brass" appear within `cite`?
- `dfn` can be useful for producing an index.
- These elements are defined as follows:

```
<!ENTITY  % phrase   "em | strong | dfn | code |
                      samp | kbd | var | cite |
                      abbr | acronym" >
<!ELEMENT (%phrase;) - - (%inline;)*>
<!ATTLIST (%phrase;) %attrs;>
```

- These elements are themselves inline elements.

Explicit Font Selection (1)

- One can also explicitly request certain fonts (physical font specifications).
- However, this is presentation-oriented markup. If possible, one should prefer the logical font specifications.

  Logical font specifications give the marked phrase a meaning that does not only refer to the output on paper or on the screen.
- If a font is selected only for for a nicer look, but has no meaning for the content, one should use stylesheets.

Explicit Font Selection (2)

- The following fonts can be used in HTML 4 strict:
  - ◇ `tt`: `Teletype` (characters have fixed width).
  - ◇ `i`: *italics*.
  - ◇ `b`: **boldface**.
  - ◇ `big`: large font.
  - ◇ `small`: small font.
- In HTML 3.2 (and in HTML 4 transitional) there are in addition:
  - ◇ `u`: underlined text.
  - ◇ `s`, `strike`: ~~strike-through text~~.

Explicit Font Selection (3)

- The elements for the physical font selection are defined in the same way as the elements for logical font selection:

```
<!ENTITY % fontstyle "tt | i | b | big | small">
<!ELEMENT (%fontstyle;) - - (%inline;)*>
<!ATTLIST (%fontstyle;) %attrs;>
```

- It is possible to nest font selection elements.
  Some browsers will display e.g. bold italic, some will simply choose the innermost font.

Explicit Font Selection (4)

- In HTML 3.2 (and HTML 4.01 transitional), there is in addition an element `font` with the attributes:
  - ◇ `size`: Size between 1 and 7, also relative, e.g. `+1`.
  - ◇ `color`: Color, e.g. `#FF0000` or `red`.
  - ◇ `face`: List of font names in the sequence of preference, separated by “,”

    The attribute `face` was not officially permitted in HTML 3.2, but it is contained in HTML 4.01 transitional. The browser chooses the first font on the list that it has.
- In addition there is `basefont`, to set the normal character size. It has only the single attribute `size`.

Exponents/Indices: sup, sub

- `sub` (“subscript”) is used to mark indices.
  E.g. x₂ gives x_2.
- `sup` (“superscript”) is used to mark exponents.
  E.g. x² gives x^2.
- But x₂³ gives $x_2{}^3$, not x_2^3.
- These elements are defined like the other font specifications:

```
<!ELEMENT (sub|sup) - - (%inline;)*>
<!ATTLIST (sub|sup) %attrs;>
```


Short quotes: q (1)

- HTML 4.01 has an element `q` for marking quotations that are displayed within the running text.

  Actually, `q` appeared already in very old HTML versions, but it is missing in the HTML 3.2 standard.
- `blockquote` is used for long quotations that appear as a paragraph of their own (see below).
- The content of the element `q` is automaticaly displayed within quotation marks.

  It is wrong to write explicit quotes. Note that the right quotation marks depend on the language. Also nested `q`-elements are possible, the browser may use e.g. single and double quotes.

Short quotes: q (2)

- The element type `q` is defined like the font specifications, but has an additional attribute "`cite`":

```
<!ELEMENT q - - (%inline;)*>
<!ATTLIST q %attrs;
            cite %URI; #IMPLIED>
```

- With the attribute `cite` one can refer to the cited document (if it has an URI).
- `q` belongs to the inline elements.

Container for Stylesheets: span

- With the element `span` one can mark text pieces in order to refer to them e.g. in stylesheets.

  E.g. if one wants to display text in a different font or color, but none of the logical phrase elements (like `em`) fits (has the right meaning). Another application is a "Tool-Tip", which one can define for a certain piece of text with the attribute `title`.

- Without stylesheet, the text is displayed normally.
- `span` (new in HTML 4) is defined as follows:

```
<!ELEMENT span - - (%inline;)*>
<!ATTLIST SPAN %attrs;>
```

- `span` is an inline element.

Summary: Inline Elements

- HTML 4.01 has 31 inline-elements classified as:
  - ◇ Logical font specification: `em`, `strong`, `dfn`, `code`, `samp`, `kbd`, `var`, `cite`, `abbr`, `acronym`.
  - ◇ Physical font specification: `tt`, `i`, `b`, `big`, `small`.
  - ◇ Objects in forms: `input`, `select`, `textarea`, `label`, `button`.
  - ◇ Other elements: `a`, `img`, `object`, `br`, `script`, `map`, `q`, `sub`, `sup`, `span`, `bdo`.
- The entity `inline` is defined as text (`#PCDATA`) plus all these elements.

Overview

Paragraphs: `p` (1)

- Paragraphs contain the normal text of the document (i.e. text that is not a headline etc.).

  There are also other containers for text, e.g. the element `li` (list item). It is not necessary to enclose the text of a list item in a `p`-element.

- One cannot write text directly in the `body` element. It must be enclosed in a block-level element like `p`.
- Paragraphs are normally displayed by ending the currently line, putting some empty vertical space, and beginning a new line for the text inside the paragraph.

Paragraphs: `p` (2)

- The element type `p` is declared as follows:

```
<!ELEMENT p - O (%inline;)*>
<!ATTLIST p %attrs;>
```

- The start tag is mandatory, but not the end tag.
- Within the `p` element, one can write a (possibly empty) sequence of text pieces and inline elements (e.g. `tt`, `a`).
- `p` is a block-level element.

Headlines: `h1`, . . . , `h6` (1)

- Headlines of different levels are marked with the elements `h1` to `h6`, e.g.
  - ◇ `h1` are headlines of chapters,
  - ◇ `h2` are headlines of sections within chapters,
  - ◇ `h3` are headlines of subsections, etc.
- Headlines are block-level elements.

  E.g. a headline cannot be contained within a paragraph. Since the end-tag of `p` is optional, this is normally no problem: The HTML parser simply closes the current paragraph if it sees the start tag of a headline. But e.g. `<em><h1>...</h1></em>` is a syntax error.

Headlines: `h1`, . . . , `h6` (2)

Example:

```
<h1>First Chapter</h1>
<p>Introduction of the first chapter
<h2>First Section of the First Chapter</h2>
<p>...
<p>...
<h2>Second Section of the First Chapter</h2>
<P>...
<h1>Second Chapter</h1>
<h2>First Section of the Second Chapter</h2>
<p>...
```

Headlines: h1, . . . , h6 (3)

- The structure of the headlines should be logical:
  - ◇ The first headline should be of type h1.
  - ◇ After a headline of type hi, the next headline should have type hj with $j \leq i + 1$.

    Since in Netscape, h2 headlines are as large as h2 headlines, one might be tempted to jump from h1 directly to h3. However, this is bad style. If necessary, one can change the font size with a stylesheet.
- Currently, browsers do not automatically generate chapter/section numbers.

  With future CSS versions, it should be possible to write stylesheets that do this.

Headlines: h1, . . . , h6 (4)

- Advanced browsers may generate a table of contents from the headline elements.
- Search engines may give higher weight to words in headlines.
- It would have been better to create elements for chapters, sections, etc. instead only elements for chapter/section headlines.

  With the current elements, it is not simple to select the contents of a section. E.g. a h2 section ends with the next h2 or h1 element, or with the end of the document. One can use the element div to include a headline together with the text of the corresponding section.

Headlines: h1, . . . , h6 (5)

- The headline elements are declared as follows:

```
<!ENTITY % headline "h1|h2|h3|h4|h5|h6">
<!ELEMENT (%headline;) - - (%inline;)*>
<!ATTLIST (%headline;) %attrs;>
```

- Headlines are block-level elements. They can only contain text and inline elements.

  In HTML 3.2 (and HTML 4.01 Transitional), h1-h6 have an attribute align. It can take the values left, center, right, justify.
- Headlines are normally printed boldface, the font size decreases from h1 to h6, and there is some empty vertical space above and below the headline.

Contact Address: address (1)

- Contact information for the document, e.g. the email address of the author, can be given in the element address, e.g.

```
<address>Stefan Brass
    (<a href="mailto:brass@acm.org"
        >brass@acm.org</a>)</address>
```

- address may be useful for future search engines.
- Browsers may display the text marked with address in a special information window.

  Unfortunately, there is no rule for the syntax of the contents of address elements. Therefore, it is difficult to automatically evaluate it.

Contact Address: `address` (2)

- Often the `address` element also contains the date of last change, but this is not its real purpose.
- Often the `address` element is used at the very bottom of the document, sometimes also at the beginning.
- Normally, the contents of `address` is displayed in italics in its own paragraph.

Contact Address: `address` (3)

- `address` is declared as follows (in the same way as the headlines `h1`-`h6`):

```
<!ELEMENT address - - (%inline;)*>
<!ATTLIST address %attrs;>
```

- In HTML 3.2, headlines and `address` could only be used as direct children of `body` (not nested within other elements).
- In HTML 4.01 they are normal block-level elements (this weakens the structure of HTML).

Delimiting Lines: `hr`

- `hr` ("horizontal rule") can be used to draw a line in order to separate pieces of the document.

  However, this is also at least in part presentation-oriented.

- Example:

```
<p>
<hr>
<h2>Headline</h2>
<hr>
```

Headline

- `hr` is declared as follows:

```
<!ELEMENT hr - O EMPTY>
<!ATTLIST hr %attrs;>
```


Item Lists: `ul`, `ol` (1)

Example for an unordered list: HTML contains ...

- Unordered lists: Items are marked by bullets (`ul`)
- Ordered lists: Items are numbered (`ol`)
- Definition lists: Items are marked by words (`dl`)

Example for an ordered list: HTML contains ...

1. Unordered lists (`ul`)
2. Ordered lists (`ol`)
3. Definition lists (`dl`)

   Definition lists are treated separately here, see below.

Item Lists: ul, ol (2)

- ul and ol are declared as follows:

```
<!ELEMENT ol - - (li)+>
<!ATTLIST ol %attrs;>
<!ELEMENT ul - - (li)+>
<!ATTLIST ul %attrs;>
```

- ul and ol are block elements. They must contain a non-empty sequence of li ("list item")-elements. Start and end tag are required.

  In HTML 3.2 ul and ol had the attributes type (disc, square, circle for ul and 1, a, A, i, I for ol), start (only OL) and compact. In addition, there were list types dir und menu.

Item Lists: ul, ol (3)

- li is declared as follows:

```
<!ELEMENT li - O (%flow;)*>
<!ATTLIST li %attrs;>
```

- The contents of li ist %flow;, that is the union of block elements, inline elements, and text. Only the start tag is necessary, the end tag can be left out.
- Since block elements are allowed inside li, lists can be nested.
- Usually, list elements are displayed slightly indented.

Definition Lists: dl (1)

- Definition lists consist of a sequence of entries, where each entry consists of a word and some text. Typical application: Glossary.
- Example:

  Unordered List:
  : The list items are marked with a symbol like "•".

  Ordered List:
  : The list items are numbered.

  Definition List:
  : Each item is marked with a word.

Definition Lists: dl (2)

- Within the element dl, the elements dt ("definition term") and dd ("definition description") are used:

```
<dl>
<dt>Unordered lists:
<dd>The list items are marked with symbols like
    e.g. bullets.
<dt>Ordered lists:
<dd>The list items are numbered.
</dl>
```

- The normal sequence is dt, dd, dt, dd, . . . , but other sequences are legal.

DL: Definitions-Listen (3)

- The definition list elements are declared as follows:

```
<!ELEMENT DL - - (DT|DD)+>
<!ATTLIST DL %attrs;>
<!ELEMENT DT - O (%inline;)*>
<!ELEMENT DD - O (%flow;)*>
<!ATTLIST (DT|DD) %attrs;>
```

- `dl` is a block element and contains a non-empty sequence of `dt` and `dd`-elements.
- `dt` can only contain inline-elements, but `dd` can contain inline and block-elements (plus text).

  Thus, definition lists can be nested. HTML 3.2 had a bolean attribute `compact` (prints term and description in the same line if possible).

Citations: blockquote (1)

- Longer citations that consist of one or more paragraphs can be enclosed in the `blockquote`-element.

  The element `q` is intended for shorter citations, see above.

- Example:

```
<blockquote cite="http://bible.gospelcom.net/">
<P>Denn so hat Gott die Welt geliebt,
daß er seinen eingeborenen Sohn gab,
damit jeder, der an ihn glaubt, nicht verloren
gehe, sondern ewiges Leben habe.
</blockquote>
```

- One can reference the cited resource with the optional attribute `cite` (new in HTML 4).

Citations: blockquote (2)

- `blockquote` is defined as follows:

```
<!ELEMENT blockquote - - (%block;|SCRIPT)+>
<!ATTLIST blockquote %attrs;
                     cite %URI; #IMPLIED>
```

- `blockquote` is a block element, and must contain block elements (e.g. `p`).

  Text directly inside `blockquote` is not allowed.

- `blockquote` is usually displayed by indenting the enclosed paragraphs.

  Therefore, `blockquote` is used relatively often to indent part of the text, even though the indented text is no citation. That is wrong and one should use stylesheets.

General Container: div (1)

- `div` ("Division") is like `span` a container-element without semantics of its own.
- Whereas `span` is an inline element, `div` is a block element.
- Before and after a `div`-Element, there is normally a line break.

  In contrast, `span` (without stylesheet) does not make any difference for the display of the text. Block elements use a number of lines in the browser window exclusively, therefore the behaviour of `div` is natural.

- One can mark part of the text with `div`/`span`, e.g. in order to reference it in style sheets.

General Container: div (2)

- One can use div/span in order to simplify the extraction of certain information pieces from HTML:

```
<div class="person">
<span class="name">Brass</span>:
<span class="phone">32150</span>
</div>

<div class="person">
<span class="name">Kroeger</span>:
<span class="phone">32140</span>
</div>
```


General Container: div (3)

- div is declared as follows:

```
<!ELEMENT div - - (%flow;)*>
<!ATTLIST div %attrs;>
```

- div-elements can contain block elements and inline elements.

In HTML 3.2, div had an attribute align, with the possible values left, right, center. In addition, there was an element center, that was equivalent to <div align=center>. Thus, the contents of the element was printed centered on the page.

Summary: Block Elements

HTML 4 has 19 block elements:

- Paragraph: p.
- Headlines: h1, h2, h3, h4, h5, h6.
- Lists: ul, ol, dl.
- Preformatted Text: pre.
- Contact Information: address.
- Citations: blockquote.
- Delimiting Lines: hr.
- Other: div, noscript, form, table, fieldset.