Chapter 9: HTML/XHTML II

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- NCSA Beginner's Guide to HTML (no longer maintained).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.

Overview

Hypertext-Links: `a` (1)

- The Web is a hypertext/hypermedia system.
- I.e. it is not a set of single, isolated documents, but it is a directed graph with
  - ◇ documents as nodes,
  - ◇ links between documents as edges.
- "Although a simple concept, the link has been one of the primary forces driving the success of the Web." [HTML 4.01 Specification]

Hypertext-Links: a (2)

- A link has two ends, which are called "anchors". The link refers from the "source anchor" to the "destination anchor".
- With elements of type `a` and the attribute `href` one can put a source anchor (reference to another document) into the text of a document:

```
See also: <a href="http://www.w3.org">W3C</a>.
```

- Displayed as (see below for details): `See also: W3C.`
- The text "`W3C`" is called the label of the link.

Hypertext-Links: `a` (3)

- The destination anchor can be any document in the world wide web. It does not have to be an HTML document (e.g. also PDF file, image, program).
- This shows the integrating power of HTML, which was an important success factor for the WWW.

  It is not even necessary that the goal document is fetched via the HTTP protocol, although this is today the most common case.
- If the link destination is an HTML document, one can not only refer to the document as a whole, but also to a destination anchor inside it (see below).

Hypertext-Links: `a` (4)

- Hypertext-links (source anchors) are normally displayed underlined and in a different color.
- There are usually two different colors for links pointing to documents that have already been visited and documents that have not yet been visited.

  Typically, links that have not yet been visited are colored (light) blue, and links that have already been visited are colored red, purple, or brown. Sometimes, there is a third color for links that have just been clicked on (displayed while the document is fetched). These colors can easily be changed, but that is considered bad style, since the color codes important information for navigating in the web. Of course, the browser remembers only for some time whether a page was visited.

Hypertext-Links: `a` (5)

- When the user moves the mouse pointer on a link, most browsers show the referenced URI in the bottom line of the browser window.

  In addition, the form of the mouse pointer changes.

- With the attribute `title` of the `a`-element, one can also specify a text that pops up as a tool tip when the mouse pointer stays some time over the link.

  This might help users who still think about clicking on the link. Old browsers do not have this feature, it became available in Internet Explorer 4.0.

Hypertext-Links: `a` (6)

- What happens when the user activates a link (clicks on it), depends on the URI, the media type of the referenced resource, and the browser configuration:
  - ◇ HTML documents, ASCII text files, images, etc. are normally displayed in the same browser window (they replace the current document).
  - ◇ The browser can be configured in such a way that it calls helper applications to display certain media types (e.g. `ghostview` for postscript).
  - ◇ It is also possible to store the data in a file.

Hypertext-Links: a (7)

- Possible reactions on link activation, continued:
  - ◇ When a user clicks on a `mailto:`-URL, an email program is called.

    In the same way, a `telnet`-window can be opened (for `telnet:` URLs), a news reader (`news:`, `nntp:`), etc.
  - ◇ One can specify that the document should be displayed in a new window (see frames below):

```
<a href="http://www.w3.org" target="_blank"
        >World Wide Web Consortium</a>
```

  - ◇ It is also possible that a Javascript program is executed when the user clicks on a link.

Hypertext-Links: a (8)

- Instead of clicking on links with the mouse, one can use the Tab-key to jump from one link to the next, and then activate the link with Enter/Return.

  This of course depends on the browser. If a link is selected with the Tab-key, it gets the focus. Shift-Tab jumps back to the previous link. Backspace goes back to the previous page.

- The element type a has an attribute tabindex for specifying the sequence in which Tab selects the links. (e.g. tabindex="2").
- Furthermore, one can specify with accesskey="A" that this link should be selected with Alt-A.

Hypertext-Links: `a` (9)

- Links can also contain information about the referenced document:
  - ◇ `type`: Media type, e.g. `text/html`.
  - ◇ `hreflang`: Language, e.g. `en-US`.
  - ◇ `charset`: Character encoding, e.g. `ISO-8859-1`.

    A complete list of registered values is available at [http://www.iana.org/assignments/character-sets]
- Normally, one gets this information when the referenced document is accessed. But see next slide.

Hypertext-Links: `a` (10)

- Reasons why information (meta data) about the referenced document in the link might be useful:
  - ◇ If a search engine cannot use the media type, it should not even follow the link.
  - ◇ If the resource is accessed via FTP (not HTTP), one does not get this information.
  - ◇ If the server of the referenced document does not set `Content-Language`, this information can be built into the link.

Hypertext-Links: `a` (11)

- It is also possible to describe the relationship between this document and the referenced document, e.g. `chapter1.html` might contain:

```
<a href="chapter2.html"
rel="Next">Chapter 2</a>
```

- The attributes `rel` and `rev` are explained in more detail for the `link`-element below.

  Knowing such relationships might be useful for advanced navigation tools and future search engines. They are also important in order to download all pages of a structured document and to print them in the right sequence. See below.

Hypertext-Links: a (12)

- The element type a is declared as follows:

```
<!ELEMENT a - - (%inline;)* -(a)>
```

- a is an inline element and contains inline elements.

  a-elements cannot contain block elements like e.g. h1 to h6.

- a-elements cannot be nested.
- In HTML 3.2, a had the attributes name, href, rel, rev, title. In HTML 4.01, it has 13 attributes (see below) in addition to the 16 standard attributes.

  In HTML 4.01 Transitional, there is still one more attribute: target.

Hypertext-Links: a (13)

```
<!ATTLIST a %attrs;
            charset   %Charset;      #IMPLIED
            type      %ContentType;  #IMPLIED
            name      CDATA          #IMPLIED
            href      %URI;          #IMPLIED
            hreflang  %LanguageCode; #IMPLIED
            rel       %LinkTypes;    #IMPLIED
            rev       %LinkTypes;    #IMPLIED
            accesskey %Character;    #IMPLIED
            shape     %Shape;        rect
            coords    %Coords;       #IMPLIED
            tabindex  NUMBER         #IMPLIED
            onfocus   %Script;       #IMPLIED
            onblur    %Script;       #IMPLIED>
```

Destination Anchors: a (1)

- A link may reference not only a document as a whole, but also a specific position within it.
- This position is marked with elements of the same type `a`, but with the attribute `name`, e.g.:

```
<h1><a name="dns">2. Domain Name System</a><h1>
```

- While the browser displays source anchors (links) normally underlined and in a different color, such destination anchors are not specially marked in the output.

Destination Anchors: a (2)

- In order to reference the marked position, one adds the name as a “fragment identifier” to the document URL (separated by “#”):

```
<a href="http://www.x.y/#dns">Chapter 2</a>.
```

- If the destination anchor is in the same document (i.e. this is a reference within the document), the empty URL can be used:

```
<a href="#dns">Chapter 2</A>.
```

Destination Anchors: a (3)

- Newer browsers do not need an a-element in order to mark a position in the text: One can add the attribute id to arbitrary body elements:

```
<h1 id="dns">2. Domain Name System<h1>
```

- The value of the attribute id can then be used as a "fragment identifiers" in the same way as the value of the name-attribute.

  The value of the attribute id must start with a letter, and after that, it can contain letters, digits, ".", ":", "-" and "_". In contrast, the attribute name is declared as CDATA (arbitrary string).

Destination Anchors: `a` (4)

- Only the element type `a` has the attribute `name`. Older browsers do not understand the `id`-attribute.
- XHTML uses only the attribute `id` to declare destination anchors.
- For maximal portability, one should use the element `a` with both attributes `name` and `id` set to the same value:

```
<a name="dns" id="dns">2. Domain Name System<a>
```

Destination Anchors: a (5)

- The values of attributes `name` and `id` must be unique in the entire document.

  In order to ensure that fragment identifiers refer to a unique element, it is also not allowed that the same value is used for an attribute `name` of one element, and as an attribute `id` for another element. If an element has both attributes `name` and `id`, the value must be the same. In this way, every element has only one name.

- It is also not allowed to use the same name with different capitalization, i.e. `id="xyz"` and `id="XYZ"` in the same document would be an error.

  On the other hand, references must use the correct capitalization. That is not logical.

Destination Anchors: `a` (6)

- An element of type `a` can be source anchor (`href`) and destination anchor (`name`, `id`) at the same time.

  It is legal to use an element of type `a` without all three attributes `href`, `name`, `id`. A script might later set these attributes.

- Elements of type `a` should not be empty.

  If a source anchor has empty contents, there is nothing on which the user can click. Empty destination anchors would make sense, but some browsers do not understand them.

The Base URI (1)

- In the document head, an element of type `base` can be used to define an URI, with respect to which all relative URIs in the document are evaluated.
- E.g. in the document "`http://www.x.com/a/b.html`" the URI "`c.html`" normally refers to the document "`http://www.x.com/a/c.html`".
- However, if the document head contains

```
<base href="http://www.y.de/d.html" />
```

the relative URI means "`http://www.y.de/c.html`".

The Base URI (2)

- E.g. when a HTML document is sent via email, the document itself has no URI, so that if one should use relative URIs in this case, “`base`” is required.
- Defining a base URI permits to move or copy the document without moving the documents that are referenced via relative URIs within it.

  However, even if one copies the entire set of documents, relative URIs will then still refer to the old place defined with “`base`”. In this case, it would have been better not to use “`base`”.

The Base URI (3)

- The element type "base" is declared as follows:

```
<!ELEMENT base - O EMPTY>
<!ATTLIST base href %URI; #REQUIRED>
```

- The entity "%URI;" is one of many entities that are simply defined as "CDATA", because a DTD cannot express the syntax restrictions:

```
<!ENTITY %URI "CDATA">
    <!-- a Uniform Resource Identifier,
         see [RFC2396] -->
```

- The element "base" must appear before all elements that contain relative URIs (e.g. link).

Links in the Head (1)

- With elements of the type `link`, one can describe relationships of this document to other documents.
- In contrast to elements of type `a` (normal hyperlinks within the text), `link`-references do not appear in the document window.
- A browser may make them available via a toolbar, a menu, or a navigation window.
- But at the moment most browsers ignore them (with the exception of the link to a stylesheet).

Links in the Head (2)

- Example: `chapter2.html` may contain the following:

```
<link rel="Start" href="start.html" />
<link rel="Contents" href="contents.html" />
<link rel="Next" href="chapter3.html" />
<link rel="Prev" href="chapter1.html" />
```

- Search engines can use the link to the start document to refer to it instead of/in addition to many single sections that all contain the search term.
- Tools that display the structure of a website can use the information contained in these links.

Links in the Head (3)

- A browser can use these links to fetch e.g. the "`next`" page while the user still reads this page ("prefetching").
- Links are also important for downloading all web pages that belong to a complex document (for off-line reading).
- A link can refer to a version of the document that is intended for printing: A browser could automatically fetch it when the user wants to print the document.

Links in the Head (4)

- E.g. a link to a print version can look as follows:

```
<link rel="alternate" media="print"
      title="Postscript-Version"
      type="application/postscript"
      href="http://www.x.com/doc.ps" />
```

- One can also refer to versions in other languages:

```
<link rel="alternate" title="English Version"
      type="text/html" hreflang="en"
      href="http://www.x.com/eng.html" />
```

Links in the Head (5)

- The HTML 4.01 specification mentions the following kinds of links: `Alternate`, `Stylesheet`, `Start`, `Next`, `Prev`, `Contents`, `Index`, `Glossary`, `Copyright`, `Chapter`, `Section`, `Subsection`, `Appendix`, `Help`, `Bookmark`.

  The HTML 3.2 specification mentions: `top`, `contents`, `index`, `glossary`, `copyright`, `next`, `previous`, `help`, `search`. In the DTD for HTML 4.01 and XHTML, a comment mentiones `start`, `contents`, `previous`, `next`, `index`, `end`, `help`, `stylesheet`, `script`, `alternate`. The HTML 4.01 specification states that capitalization is not relevant, but probably one should use all lowercase for XHTML.

  It was once recommended to put the author's email address in this way into the document: `<link rev="made" href="mailto:brass@acm.org" />`

Links in the Head (6)

- The DTD permits arbitrary strings:

```
<!ENTITY % LinkTypes "CDATA">
```

  Browsers ignore unknown link types. If one wants to use an extended set of link types, one should refer with the attribute `profile` of the `head` element to a meta data definition.

- The attribute `rel` permits a list of relationship types (separated by spaces).
- There can be several `link` elements with the same relationship type.

  E.g. the main document may contain for every chapter a link with `rel="Chapter"`.

Links in the Head (7)

- With the attribute `rev` relationship types can also be used in the opposite direction.
- A link from document A to document B with `rev=`t means the same as a link from B to A with `rel=`t.
- E.g. an index to a document could contain a link with `rev="Index"` to that document.
- Browsers cannot know about links defined in other documents.

  It is probably best to make all these links bi-directional by specifying them in both documents (in one with `rel`, in the other with `rev`).

Links in the Head (8)

- The attribute `media` is useful for the specification of variants for special devices (`rel="alternate"`). It contains a comma-separated list of these values:
  - ◇ `screen`,
  - ◇ `tty` (fixed pitch character grid),
  - ◇ `tv` (low resolution, limited scrolling), `projection`,
  - ◇ `handheld` (small screen, monochrome, . . .),
  - ◇ `print` (paged),
  - ◇ `braille`, `aural` (speech synthesizers),
  - ◇ `all` (suitable for all devices).

Links in the Head (9)

- The element type "link" is declared as follows:

```
<!ELEMENT link - O EMPTY>
<!ATTLIST link %attrs;
               charset  %Charset;      #IMPLIED
               href     %URI;          #IMPLIED
               hreflang %LanguageCode; #IMPLIED
               type     %ContentType;  #IMPLIED
               rel      %LinkTypes;    #IMPLIED
               rev      %LinkTypes;    #IMPLIED
               media    %MediaDesc;    #IMPLIED>
```

- Elements of type `link` can only appear in the `head`.

Overview

Images: `img` (1)

- Images (pictures) can be embedded into an HTML page with elements of the type "`img`".
- The images are not specified in HTML itself, but in graphics formats like `GIF`, `JPEG`, `PNG`.
- The data of the image are not contained in the HTML file, but in separate files (one per image).
- The HTML file only contains the URIs of the pictures. The browser will load the files referenced in `img`-elements automatically and display the image in the same window as the HTML text.

Images: `img` (2)

- The `img` element is empty and contains the URI of the picture in the attribute `src`:

```
<img src="brass.jpg"
     alt="Photo von Stefan Brass" />
```

- Of course, it is also possible to reference the picture with an `a` element. Differences to `img` are:
  - ◇ The browser will not load the image automatically, only when the user clicks on the link.
  - ◇ When the picture is loaded, it will replace the current document in the browser window.

Images: `img` (3)

- The attribute `alt` contains a short text which is displayed when the picture itself cannot be shown:
  - ◇ For blind persons, speech generators.
  - ◇ If the browser does not understand the graphics format of the image.
  - ◇ If the image file cannot be accessed.

    E.g. because the URL is no longer current.
  - ◇ When the user has switched off the display of images (e.g. for slow internet connections).
  - ◇ For search engines.

Images: `img` (4)

- In HTML 3.2, `alt` was optional. In HTML 4, it is required.

  For pictures that do not contain information and are only intended to make the web page more beautiful, the HTML specification recommends `alt=""`.

- In HTML 4, `img` has in addition an optional attribute `longdesc`, which contains the URI of a longer description of the picture.

  E.g. a blind person can select this if the `alt`-text sounds interesting or does not contain enough information.

Images: `img` (5)

- It is recommended to specify the height and width of the picture (in pixels) in the `img` element:

```
<img src="brass.jpg" alt="S. Brass"
     width="256" height="192" />
```

- In this way, the browser can reserve sufficient space for the picture in the window when it displayes the HTML text.

  Of course, the image file itself contains this information, but the browser gets it only after the HTML file. It displayes the HTML text, even if it does not know yet how much space the picture will need. However, if the picture needs more space, the HTML text must be drawn again with e.g. different line breaks.

Images: `img` (6)

- The attributes `width` and `height` can also be used in order to modify the size of the picture.

  The browser than tries to scale the picture. However, making the picture smaller than its real size wastes network bandwidth.

- `img` can be used inside `a` (it is an inline element).

  Browsers usually mark clickable pictures with a colored border.

- It is quite common to link a small version of a picture to a larger version (with higher resolution):

```
<a href="big.jpg"><img src="small.jpg" .../></a>
```

Images: `img` (7)

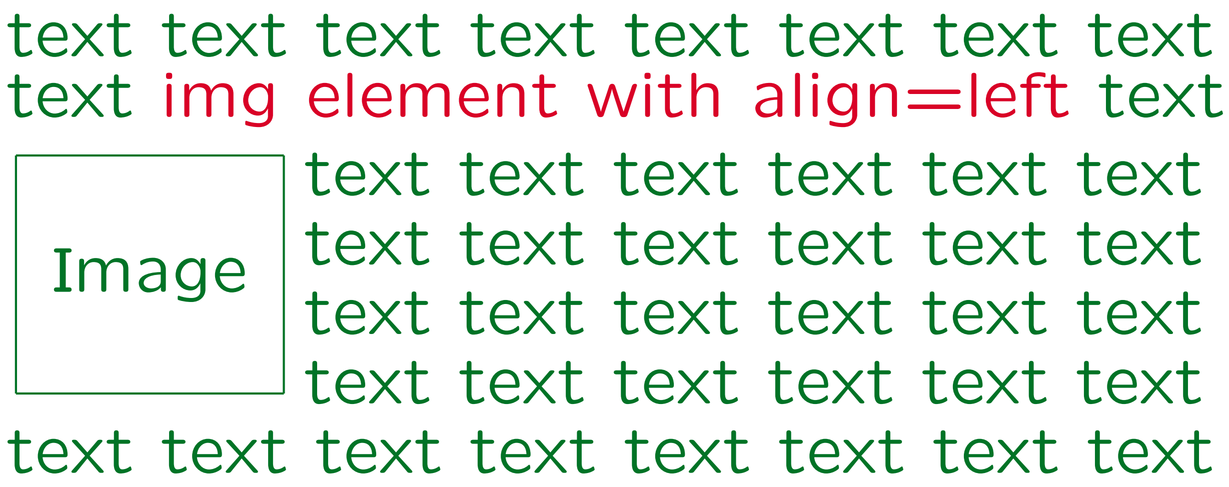
- If an `img` element appears within a text line, its lower border is aligned with the base line of the text.

  `xyz xyz` $\boxed{\texttt{Picture}}$ `xyz xyz`

- In HTML 3.2/HTML 4 Transitional, `img` has an attribute `align`:
  - ◇ `align="bottom"`: This is the default (see above).
  - ◇ `align="top"`: Upper border of the picture = upper border of the current text line.
  - ◇ `align="middle"`: The center of the picture is aligned with the base line of the surrounding text.

Images: `img` (8)

- With the above settings for `align`, the picture is simply treated like a big letter.
- However, with `align="left"` or `align="right"`, one can move the picture to the left or right side of the window and let the text "flow around it":



text text text text text text text text
text img element with align=left text
text text text text text text
text text text text text text
text text text text text text
text text text text text text
text text text text text text text text

Images: `img` (9)

- In HTML 3.2/HTML 4 Transitional,

  ```
  <br clear="left" />
  ```

  can be used to move the writing position below the picture on the left side.

- In HTML 4 Strict, `img` has no attribute `align`, and `br` has no attribute `clear`. One must use stylesheets instead.

- Often, pictures are also used together with tables.

Images: img (10)

- Furthermore, img had the following attributes in HTML 3.2/HTML 4 Transitional:
  - ◇ hspace: Horizontal empty space to the left and right of the picture.
  - ◇ vspace: Vertical empty space above and below the picture.
  - ◇ border: Size of the border that the browser draws around clickable images.

    E.g. with border="0", this border is not displayed.

Images: `img` (11)

- In HTML 4 (Strict and Transitional), `img` has an attribute `name`, in order to reference the picture in stylesheets and scripts (programs). However, one should use `id` instead.

  The attribute `name` is kept only for compatibility reasons (although it was never officially part of HTML 3.2). It cannot be used for destination anchors (at least, it is not intended for that purpose).

- Finally, there are two further attributes: `usemap` and `ismap` (in HTML 3.2, HTML 4 Strict+Transitional). They are used for “image maps” (see below).

Images: img (12)

- The element type img is declared as follows:

```
<!ELEMENT img - O      EMPTY>
<!ATTLIST img %attrs;
              src      %URI;     #REQUIRED
              alt      %Text;    #REQUIRED
              longdesc %URI;     #IMPLIED
              name     CDATA     #IMPLIED
              height   %Length;  #IMPLIED
              width    %Length;  #IMPLIED
              usemap   %URI;     #IMPLIED
              ismap    (ismap)   #IMPLIED>
```

- img is an inline element.

Image Maps

- "Image maps" make it possible that the user reaches different documents depending on the point or region in the picture where he/she clicks.
- There are two kinds of image maps:
  - ◇ Server-Side Image Maps: The browser tells the server the mouse coordinates. The server delivers different data based on these coordinates.
  - ◇ Client-Side Image Maps: The HTML file contains a list of regions in the picture, each with its own URI. The browser loads the selected URI.

Server-Side Image Maps (1)

- Server-side image maps are encoded by using an `img`-element directly enclosed in an `a`-element, where the `img`-element has the boolean attribute `ismap`:

```
<a href="http://www.x.com/get.pl">
<img src="http://www.x.com/map.gif"
     ismap="ismap" alt=... />
</a>
```

- In HTML, one would simply write `ismap` instead of `ismap="ismap"`. However, XML (and thus XHTML) require always an attribute value.

  "`ismap`" is the only possible value of the attribute (besides a "NIL" value when the attribute is not specified).

Server-Side Image Maps (2)

- If the user clicks e.g. at position (10,20) in the picture, the browser accesses the following URL:

  `http://www.x.com/get.pl?10,20`

  I.e. the effect of the `ismap`-attribute is that the coordinates of the mouse click within the picture (relative to the upper left edge of the picture) are appended to the URL to which the picture is linked.

- In this case, a CGI-program computes a web page depending on the coordinates.
- Many web servers have built-in support for image maps. Then no programming is necessary.

Server-Side Image Maps (3)

Disadvantages of server-side image maps:

- If the picture cannot be displayed, no selection is possible.

  E.g. blind persons and search engines cannot determine which web pages are reachable. Convention: The client should send coordinates 0,0 in this case, server should answer with list of links.

- If the user has clicked on a non-active part of the picture, nevertheless a request is sent.

  Only the server knows the map, and which parts on it are active.

- Unnecessary load on the server (additional task).
- Server must run: Page cannot be read offline.

Client-Side Image Maps (1)

- Client-side image maps are defined in a `map`-element that is referenced in the attribute `usemap` of `img`:

```
<img src="..." alt="..." usemap="#my_map" />
<map name="my_map">
    <area href="chapter1.html" alt="..."
          shape="rect" coords="0,0,50,10" />
    <area href="chapter2.html" alt="..."
          shape="rect" coords="0,12,50,22" />
</map>
```

- Within the `map`-element, `area`-elements are used to specify the regions and their associated URLs.

Client-Side Image Maps (2)

- The coordinates (usually in pixels, % is also possible) are relative to the left upper edge of the picture.
- The attribute `shape` can have the following values:
  - ◇ `rect`: Rectangle, `coords` must have the form:

    `"x-left,y-upper,x-right,y-lower"`
  - ◇ `circle`: Circle, `coords` must have the form:

    `"x-center,y-center,radius"`
  - ◇ `poly`: Polygon, `coords` must have the form:

    `"x1,y1,x2,y2,...,xN,yN,x1,y1"`

Client-Side Image Maps (3)

- In addition, HTML 4 (but not HTML 3.2) permits `shape="default"`.

    This should be (must be?) the last `area`-element.

- If two regions overlap, the first `area`-element has priority.

    I.e. the browser checks the `area`-elements one by one in the specified sequence, and stops as soon as it found one that contains the mouse coordinates.

Client-Side Image Maps (4)

- In HTML 3.2, the contents of `map` had to be a sequence of `area`-elements.
- HTML 4 permits also normal block-content (paragraphs, lists, etc.) instead of the `area`-elements.
- In this case, the contents is displayed normally. It is considered an alternative to the image.
- Now `a`-elements inside the `map` have the function of the `area`-elements.

  For this reason, `a` got the additional attributes `shape` and `coords` in HTML 4. `area`-elements are ignored when the `map` contains text.

Client-Side Image Maps (5)

- `map` is declared as follows:

```
<!ELEMENT map - - ((%block;) | area)+ >
<!ATTLIST map %attrs;
              name CDATA #REQUIRED>
```

- However, it only makes sense to use either a list of block elements or a list of `area`-elements.
- `map` is an inline element.
- Every `map`-element must have a name, in order to be referenced from the `img`-element with `usemap`.

  Although `usemap` is declared as URI, one should use a document-local reference.

Clent-Side Image Maps (6)

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

```
<!ELEMENT area - O        EMPTY>
<!ATTLIST area %attrs;
               shape     %Shape;      rect
               coords    %Coords;     #IMPLIED
               href      %URI;        #IMPLIED
               nohref    (nohref)     #IMPLIED
               alt       %Text;       #REQUIRED
               tabindex  NUMBER       #IMPLIED
               accesskey %Character;  #IMPLIED
               onfocus   %Script;     #IMPLIED
               onblur    %Script;     #IMPLIED>
```

Multimedia Objects: `object`

- The element type `object` can be used in general to embed multimedia objects into the HTML text.

  `object` is new in HTML 4. Especially, it replaces `applet` (regions in the browser window in which Java programs run).

- E.g. one can use `object` instead of `img`:

```
<object data="brass.jpg" type="image/jpg">
Photo of <em>Stefan Brass</em>.
</object>
```

- The contents of the `object`-element is shown only if the object itself cannot be displayed.

Summary: Links

- Links to other documents in the world wide web occur in the following element types:
  - ◇ `a`, `area`, `link`, `base`: `href`
  - ◇ `img`: `src`, `longdesc`, (`usemap`)
  - ◇ `object`: `classid`, `codebase`, `data`, `archive`, (`usemap`)
  - ◇ `q`, `blockquote`, `ins`, `del`: `cite`
  - ◇ `form`: `action`
  - ◇ `input`: `src`, (`usemap`)
  - ◇ `head`: `profile`
  - ◇ `script`: `src`, (`for`)

Overview

General Remarks (1)

- Tables make it possible to display data in form of lines and columns:

HTML Version	**Publication Date**
HTML 3.2	14.01.1997
HTML 4.0	24.04.1998
HTML 4.01	24.12.1999

- Without special support for tables, a multi-column output would only be possible with a font of fixed width (`tt`).

  Typically inside a `pre` element. The problem is to align the second and following columns at a common edge.

General Remarks (2)

- Tables were new in HTML 3.2. There still a quite simple model was used.
- In HTML 4, the possibilities were significantly extended, e.g.
  - ◇ for page breaks within printed tables,
  - ◇ for blind users and speech generators,
  - ◇ for formatting the table while data still arrive,
  - ◇ for the selection of frames and separating lines.
- But simple HTML 3.2 tables are valid in HTML 4, too.

General Remarks (3)

- Because CSS1 stylesheets do not support tables, the HTML 4 table elements still have presentation-oriented attributes.

  With CSS2 these attributes become unnecessary.

Simple Tables (1)

- Tables are enclosed in the element `table`.
- In the simple case, the contents of `table` is a sequence of `tr`-elements ("table row").
- Each table row contains a sequence of
  - ◇ `td`-elements ("table data cell"), which describe table entries, and
  - ◇ `th`-elements ("table header cell"), which are normally used in the first table row or the first column to define headlines of the columns/rows.

    By default, `th` is displayed bold und centered, `td` is shown in a normal font, left aligned.

Simple Tables (2)

- Example:

```
<table>
    <tr><th>Course</th>
        <th>Instructor</th>
        <th>Class</th></tr>
    <tr><td>Lisp</td>
        <td>Kr&ouml;ger</td>
        <td>Wed 12-14</td></tr>
    <tr><td>WWW</td>
        <td>Brass</td>
        <td>Fri 10-12</td></tr>
</table>
```

Simple Tables (3)

- In HTML (not XHTML), the end tags of `tr`, `th`, `td` are optional, thus the table can also be written:

```
<table>
<tr><th>Course <th>Instructor  <th>Class
<tr><td>Lisp   <td>Kr&ouml;ger <td>Wed 12-14
<tr><td>WWW    <td>Brass       <td>Fri 10-12
</table>
```

- The table is displayed as follows:

Course	Instructor	Class
Lisp	Kröger	Wed 12–14
WWW	Brass	Fri 10–12

Simple Tables (4)

- Without further settings, no border is printed.
- The attribute `border` sets the width of the border around the table and between the lines and columns (default is 0):

```
<table border="3">
```

- Then every table cell is enclosed in a box:

Course	Instructor	Class
Lisp	Kröger	Wed 12–14
WWW	Brass	Fri 10–12

Simple Tables (5)

- In the screen display, tables look like a relief: The table frame rises slightly above the background.
- This effect becomes stronger if the `border`-size is increased.

  The attribute `border` defines the size of the border around the table (the part that rises from the background). Although also the lines between the table cells appear when `border` is set to something different from 0, otherwise the number only influences the outer border.

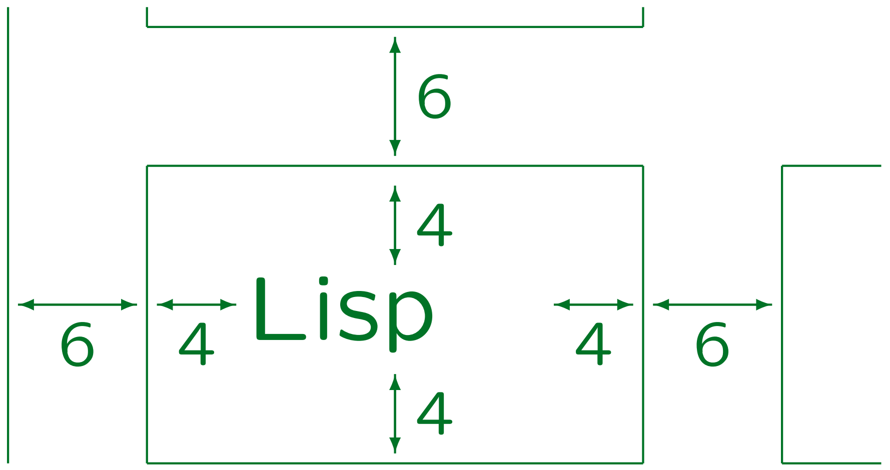
- A table cell is displayed only for non-empty entries.

  If necessary, use `&nbsp;` as table entry.

Simple Tables (6)

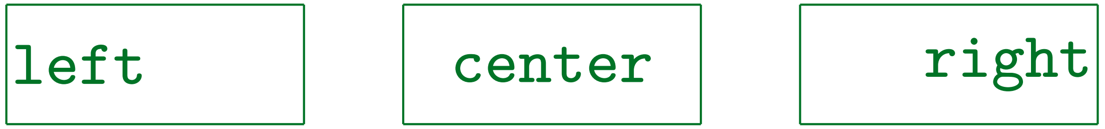
- `cellspacing` sets the empty space between the table cells (thickness of the frame/wall between cells).
- `cellpadding` sets the empty space between the frame and the cell data (margin around the cell data).

```
<table border="1" cellspacing="6" cellpadding="4">
```

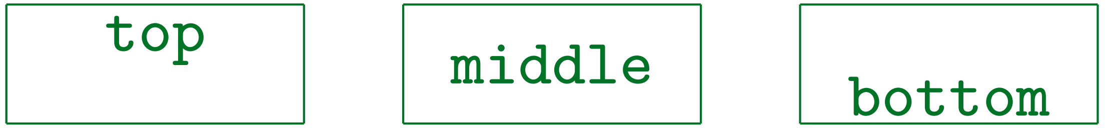


Simple Tables (7)

- The alignment of the data within the cell can be chosen with the following attributes of `td`, `th`, `tr`:
  - ◇ `align`: Horizontal alignment (`left`, `center`, `right`)



  - ◇ `valign`: Vertical alignment (`top`, `middle`, `bottom`)



- Settings specified in `tr` define the alignment for all cells in the row (unless overridden in the cell).

Simple Tables (8)

- The browser computes the width of each column.

  The following algorithm is suggested in the HTML 3.2 specification. It is not binding, the browser may choose a different algorithm.

- First, it computes the maximal width of the table entries in the column ("optimal column width").

Instructor	$\longleftarrow$ Entry with maximal width
Kröger	
Brass	

- These column widths are chosen, if they do not produce a table that is wider than the browser window (i.e. if not horizontal scroll bar is necessary).

Simple Tables (9)

- Otherwise line breaks are done in the table entries.
- Besides the optimal width of each column, also the required with is computed, which would be necessary if a line break were done after each word.

  I.e. required width = maximum over the widths of the single words in the column. In contrast, the optimal width assumes no line breaks.
- Then first every column is assigned its required width, afterwards the remaining space is distributed over the columns in proportion to the differences "optimal width − required width".

Simple Tables (10)

- One can explicitly set the width of a table:

  ```
  <table width="70%" border="1">
  ```

  The percentage is relative to the width of the browser window. Alternatively, one can specify the width in pixels (number without "%").

- In HTML 3.2/HTML 4.01 Transitional, one can set the width and height of every single table entry.

  In HTML 4.01 Strict, only the width can be chosen for entire columns (see `col`/`colgroup` below).

- In these HTML versions, one can forbid line breaks with the boolean attribute `nowrap`.

  In HTML 4.01 Strict, use `&nbsp;` or stylesheets.

Simple Tables (11)

- Within the table entries (`td`, `th`) text, inline elements, and block elements can be used (`%flow;`).

  Therefore, one can explicitly request line breaks with `br`. Also, one can enter structured text with e.g. `p`, `ul`, `ol`.

- Especially, tables can also be nested.

  It is said that this might confuse older browsers.

- Also pictures are permitted within tables.

  This is applied e.g. for using a graphics symbol in lists instead of the standard "bullet".

- In HTML 4.01 Transitional, `table`, `tr`, `td`, `th` have also an attriute `bgcolor` to set the background color.

Simple Tables (12)

- Sometimes, single table entries must extend over more than one table row or column.
- The element types `td` and `th` for table entries have attributes `colspan` and `rowspan` to select the number of columns/rows in the table grid for this entry.

Column 1	**Column 2**	**Column 3**
Large Entry: 2 Rows, 2 Columns (rowspan=2, colspan=2)		(1,3)
		(2,3)
(3,1)	(3,2)	(3,3)

Simple Tables (13)

- Table cells that are already covered by a larger entry are not specified again.
- E.g. the above table has the following structure:

```
<table>
    <tr><th>Column 1 <th>Column 2 <th>Column 3
    <tr><td rowspan=2 colspan=2> ... <td>(1,3)
    <tr><td>(2,3)
    <tr><td>(3,1) <td>(3,2) <td>(3,3)
</table>
```

- E.g. the third `tr`-element contains only “`<td>(2,3)`”, because the first two columns are covered by the large entry in the row above.

Simple Tables (14)

- Finally, one can define a table headline (table name) with a `caption`-element:

```
<table>
    <caption>Table 3: Selected Courses</caption>
    <tr>...
```

- `caption` can only contain text and inline elements.
- If it apears, it must be the first element inside `table` (before the `tr`-elements).
- The `caption`-text is displayed above the table.

  In HTML 3.2/HTML 4.01 Transional: also `align=bottom` possible.

Simple Tables (15)

- Simple tables (i.e. HTML 3.2 tables) are declared as follows:

```
<!ELEMENT table   - - (caption?, tr+)>
<!ELEMENT tr      - O (th|td)*>
<!ELEMENT (th|td) - O (%flow;)*>
<!ELEMENT caption - - (%inline;)*>
<!ATTLIST table   width       %Length #IMPLIED
                  border      %Pixels #IMPLIED
                  cellspacing %Pixels #IMPLIED
                  cellpadding %Pixels #IMPLIED>
```

Simple Tables (16)

- Declaration of simple tables, continued:

```
<!ATTLIST tr
          align   (left|center|right) #IMPLIED
          valign  (top|middle|bottom) #IMPLIED>
<!ATTLIST (th|td)
          nowrap  (nowrap)            #IMPLIED
          rowspan NUMBER              1
          colspan NUMBER              1
          align   (left|center|right) #IMPLIED
          valign  (top|middle|bottom) #IMPLIED
          width   %Length             #IMPLIED
          height  %Length             #IMPLIED>
```

HTML 4 Tables (1)

- In HTML 4, the table rows within a table can be grouped into headline (`thead`, e.g. column names), footer (`tfoot`, e.g. sums), and table body (`tbody`):

```
<table>
    <thead>
        <tr> ...
    <tfoot>
        <tr> ...
    <tbody>
        <tr> ...
        <tr> ...
</table>
```

HTML 4 Tables (2)

- E.g. a browser could keep the headline and the footer fixed while scrolling through the table.

  I.e. let only the table data scroll and keep the headline/footer always in sight.

- When printing the table, the headline and the footer could be repeated on each page.

- Neither Netscape 4.76 nor Internet Explorer 5.00 do that.

  This version of Netscape does not seem to understand the new constructs at all.

HTML 4 Tables (3)

- Note that `tfoot` must be specified before `tbody`, although the footer is printed at the bottom of the table.

  In this way, it might be possible to print the first page of a long table while data still arrive.

- It is possible to use several `tbody`-elements in order to partition the data rows into several groups.

  E.g. one can request separating lines between groups of rows, see below.

HTML 4 Tables (4)

- In HTML 4, one can declare columns before the table data. Advantages/Applications:
  - ◇ For incremental display of the table (while data still arrives), the browser must know how many columns there are and how wide they will be.
  - ◇ In HTML 3.2, it was not possible to define the alignment for an entire column.

    For numeric columns, one had to use `align=right` for every cell.
  - ◇ One can partition the columns into groups.

    E.g. one can then print vertical border lines only between groups. The groups are also useful for stylesheets.

HTML 4 Tables (5)

- Columns are declared with `col`-elements. These are empty and can have the following attributes:
  - ◇ Attributes for horizontal and vertical alignment of the table data (`align`, `char`, `charoff`, `valign`).

    `char` and `charoff` are explained below.
  - ◇ `width` for defining the column width.

    In pixels (e.g. 25), Percent (e.g.20%), etc. HTML Spec: Percent are relative to the available width for the table. Not in IE.
  - ◇ Standard attributes, e.g. `class` for stylesheets.
  - ◇ `span` to abbreviate repeated column declarations.

    E.g. `span=2` doubles the `col`-element.

HTML 4 Tables (6)

- Example:

```
<table>
    <col align=left>
    <col align=right>
    <col align=right>
    <thead>
        <tr> <th>Exercise <th>Points <th>Percent
    <tbody>
        <tr> <td>1        <td>10     <td>83%
        <tr> <td>2        <td>9      <td>75%
</table>
```

HTML 4 Tables (7)

- The columns must be defined before the table rows (and after a possible caption).
- Instead of single `col`-elements, one can also use `colgroup`-elements for defining groups of columns.

  This is important if one wants separating lines only between certain columns, and not between each two adjacent columns. Also, it simplifies the specification of many columns with similar attribute settings.
- The `colgroup`-elements then contain `col`-elements.
- If one uses column groups, all `col`-elements must appear inside `colgroup`-elements (no mixture).

HTML 4 Tables (8)

- `colgroup` has the same attributes as `col`.

  Settings in `colgroup` are inherited to the `col`-elements inside (but can be overridden in the `col`-elements).

- It is also possible that `colgroup`-elements are empty if they already contain the complete settings for the columns of the group.

  Then the number of columns should be defined with `span=...` in the `colgroup`. Otherwise it is assumed that the group consists of one column only.

HTML 4 Tables (9)

- Example (three colums: left/right/right aligned):

```
<table>
    <colgroup align="left"></colgroup>
    <colgroup><col align="right" />
              <col align="right" /></colgroup>
    <thead>
        ...
```

- If `col/colgroup` are used, no table row may have more entries than the declared number of clumns.
- Note that `col/colgroup` do not produce any output. Column headings must still be specified with `th`.

HTML 4 Tables (10)

- Declaration of the table element types:

```
<!ELEMENT table    - -
                   (caption?, (col*|colgroup*),
                    thead?, tfoot?, tbody+)>
<!ELEMENT caption  - - (%inline;)*>
<!ELEMENT thead    - O (tr)+ >
<!ELEMENT tfoot    - O (tr)+ >
<!ELEMENT tbody    O O (tr)+ >
<!ELEMENT colgroup - O (col)* >
<!ELEMENT col      - O EMPTY >
<!ELEMENT tr       - O (th|td)+ >
<!ELEMENT (th|td)  - O (%flow;)* >
```

HTML 4 Tables (11)

- In HTML 4, the attribute `align` (horizontal alignment of column data) can have two new values (besides `left`, `right`, `center`):
  - ◇ `justify`: left- and right aligned (spaces between words are made wider).
  - ◇ `char`: Column data is aligned at a character that is defined with the attribute `char`, e.g.

    ```
    <col align="char" char="." />
    ```

    Then the first position of the "." in each entry for this column is aligned. If an entry does not contain a ".", it is treated as if "." were added at the end.

HTML 4 Tables (12)

- Browsers do not have to support `align=justify` and `align=char`. (E.g. IE 5.50 does not support them.)
- With `valign=baseline` it is now possible to align the first lines of the table entries.
- Declaration of the alignment attributes:

```
<!ENTITY % cellhalign
  "align   (left|center|right|justify|char) #IMPLIED
   char    %Character;                      #IMPLIED
   charoff %Length;                         #IMPLIED">
<!ENTITY % cellvalign
  "valign  (top|middle|bottom|baseline)     #IMPLIED">
```

HTML 4 Tables (13)

- HTML 4 offers different styles for the frame around the table and the rules between rows/columns.

  In HTML 3.2, one either gets the complete set of lines, or none.

- The attribute `frame` of the `table`-element determines which part of the frame is drawn:
  - ◇ `void`: No frame.
  - ◇ `above`, `below`, `lhs`, `rhs`: Only one side.
  - ◇ `hsides`: Left and right side.
  - ◇ `vsides`: Above and below the table.
  - ◇ `box`, `border`: All four sides.

HTML 4 Tables (14)

- The attribute `rules` of `table` determines which delimiting lines inside the table are drawn:
  - ◇ `none`: No rules.
  - ◇ `groups`: Vertical lines between column groups, horizontal lines between `thead`, `tfoot`, `tbody`.
  - ◇ `rows`/`cols`: Only horizontal/only vertical lines.
  - ◇ `all`: All rules (between rows and columns).
- The default is `frame=void` and `rules=none`, except when `border` is set to a value different from 0.

  Then it is `frame=border`, `rules=all` for compatibility with HTML 3.2.

HTML 4 Tables (15)

- HTML 4 also makes tables better accessible for blind persons / persons using speech generators.
- In the attribute `summary` of `table`, one can describe purpose and structure of the table.
- There are different ways to describe the relationship between column headlines etc. and data elements:
  - ◇ Either one defines in the `th`-elements the `scope`of the headline (`row`, `col`, `rowgroup`, `colgroup`)
  - ◇ or one defines in each `td`-element with `headers` a list of `id`'s of the relevant `th`-elements.

HTML 4 Tables (16)

```
<table summary="This table contains the grades for
        your homeworks. Each row contains exercise
        number, number of points, and the percentage
        of the maximal number of points.">
    <thead>
        <tr> <th scope=col>Exercise
             <th scope=col>Points
             <th scope=col>Percent
    <tbody>
        <tr> <td>1 <td>10 <td>83%
        <tr> <td>2 <td>9  <td>75%
</table>
```

HTML 4 Tables (17)

- A speech generator would typically read the corresponding headline before each table entry, e.g.:

  Exercise: 1, Points: 10, Percent: 83%
  Exercise: 2, Points: 9, Percent: 75%

- If the column headings are too long, one can define an abbreviation with the attribute `abbr` of `th`.

- E.g., here the word "Percent" is unnecessary, because it is already contained in the data:

```
<TH scope=col abbr="">Prozent
```

HTML 4 Tables (18)

- Attributes of the element type `table`:

```
<!ENTITY % TFrame "(void|above|below|hsides|lhs|
                   rhs|vsides|box|border)">
<!ENTITY % TRules "(none|groups|rows|cols|all)">
<!ATTLIST TABLE
          %attrs;
          summary     %Text;   #IMPLIED
          width       %Length; #IMPLIED
          border      %Pixels; #IMPLIED
          frame       %TFrame; #IMPLIED
          rules       %TRules; #IMPLIED
          cellspacing %Length; #IMPLIED
          cellpadding %Length; #IMPLIED >
```

HTML 4 Tables (19)

- Attributes of element types col and colgroup:

```
<!ATTLIST (colgroup|col)
          %attrs;
          span           NUMBER         1
          width          %MultiLength; #IMPLIED
          %cellhalign;
          %cellvalign;>
```

- Attributes of thead, tfoot, tbody, tr:

```
<!ATTLIST (thead|tbody|tfoot|tr)
          %attrs;
          %cellhalign;
          %cellvalign;>
```

HTML 4 Tables (20)

- Attributes of table entries (`th`, `td`):

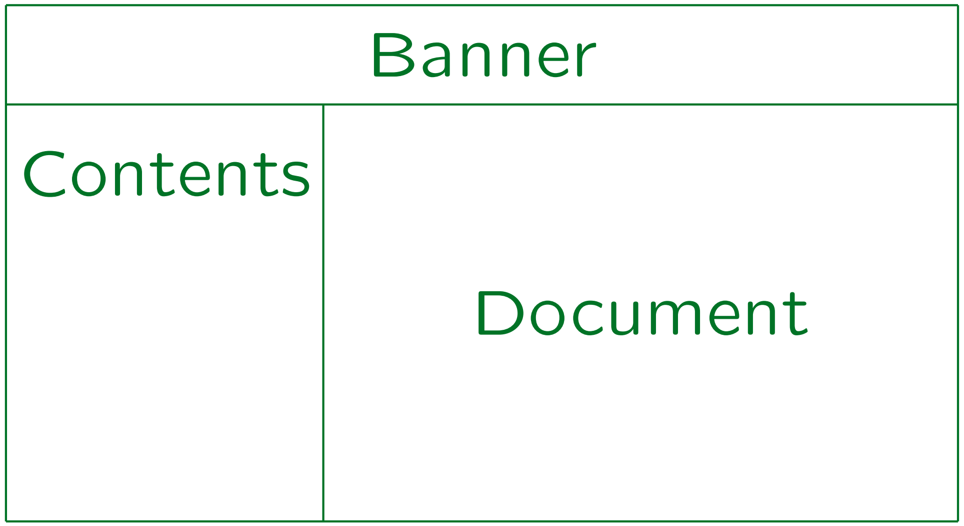
```
<!ENTITY % Scope "(row|col|rowgroup|colgroup)">
<!ATTLIST (th|td)
          %attrs;
          abbr         %Text;   #IMPLIED
          axis         CDATA    #IMPLIED
          headers      IDREFS   #IMPLIED
          scope        %Scope;  #IMPLIED
          rowspan      NUMBER   1
          colspan      NUMBER   1
          %cellhalign;
          %cellvalign; >
```

Overview

1. Hyperlinks
2. Images
3. Tables
4. Frames

General Remarks (1)

- Frames permit to split the browser window into several subwindows.
- In each subwindow (frame), a different HTML document can be displayed. Typical application:



General Remarks (2)

- In the above example, the browser displays four files:
  - ◇ the main file, which determines the division of the browser window into frames.
  - ◇ and one file each for banner, contents, and document.
- The files that define the frame contents are normal HTML and can (usually) also be viewed separately in the browser.

General Remarks (3)

- In order to view and bookmark frames separately, one must know their URLs.
  - ◇ Netscape: "Page Info" or press right mouse button in the frame, "Open Frame in Window".
  - ◇ Internet Explorer: Press right mouse button in the frame, select "Properties".
- The main file is not normal HTML: It contains an element `frameset` instead of `body`.

  Therefore, in the document type declaration at the beginning, the Frameset DTD must be referenced.

General Remarks (4)

- Frames were invented by Netscape and were already contained in Netscape Navigator 2.0.
- The HTML 3.2 standard does not contain frames, only HTML 4.0 contains them.
- Some users do not like frames:
  - ◇ Banners have little information, but need space.
  - ◇ The state of frames can change (other documents are loaded into them), but there is only one URL. Thus, bookmarks do not work.
  - ◇ Problems when printing frames.

Frame Definition (1)

- Frames are declared with the element types
  - ◇ `frameset`: Defines distribution of screen space.
  - ◇ `frame`: Defines the initial contents of a frame.
- `frameset` has the attribute `cols` and `rows`, to define a partition of the available space into columns or rows.

  Most `frameset`-elements contain only one of the two attributes. However, it is also possible to use both in order to create a grid. It is said that this might confuse old browsers.

Frame Definition (2)

- E.g. `rows="70,*"` splits the window into 70 pixels at the top (for the banner) and the rest below.
- The lower part can then be divided with a nested `frameset`-element into 20% for the contents list, and 80% for the main document: `cols="20%,80%"`
- The contents of `rows/cols` is a comma-separated list of length specifications:
  - ◇ Pixels, e.g. 70.
  - ◇ Percent of the available space, e.g. 80%.
  - ◇ Rest: `*`.

Frame Definition (3)

- It is possible to use "`*`" more than once, or to attach a factor. Then the still available space (after evaluating size specifications in pixels or percent) is divided proportionally.
- E.g. consider

  ```
  rows="100,20%,*,2*"
  ```

  and suppose that there are 500 pixels total height. Then the first three frames get 100 pixel each, and the last frame 200.

Frame Definition (4)

- For each frame, the `frameset`-element contains a `frame`-element.
- The attribute `src` of the `frame`-element contains the URI of the document that is initially shown in the frame.

  However, the contents of the frame may change. E.g. if one clicks on links in the table of contents, new documents can be loaded into the frame. See below.
- The URI often points to an HTML file, but it can also point to an image.
- The main file never contains the frame contents.

Frame Definition (5)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
        "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head><title>Example for Frames</title></head>
<frameset rows="70,*">
    <frame src="f_banner.gif" />
    <frameset cols="20%,80%">
        <frame src="f_toc.html" />
        <frame src="f_doc1.html" name="doc" />
    </frameset>
<noframes><P>Your browser does not understand frames. Sorry.
          <!-- Bad Style. Better display list of links -->
</noframes>
</frameset>
</html>
```

Frame Definition (6)

- Within `noframes`, one can define a text that is shown in browsers that do not support frames.
- Since the set of element types and attributes is constantly extended, the following behaviour of browsers is standard:
    - ◇ Unknown attributes are simply ignored.
    - ◇ Tags with unknown element types are ignored, but the contents of these element types is normally processed (printed).

Frame Definition (7)

- The frame specification is contained only in tags, there is no text. Thus it is not printed if the browser does not support frames.
- Of course, such browsers also do not understand the `noframes`-element type. Thus it ignores the tags, but it prints the contents inside.
- A browser that does understand frames also understands the `noframes` element type, and does not display its contents.

Frame Definition (8)

- Normally, the user can move the borders between the frames.

  Move the mouse pointer on the line between the frames, press the left mouse button, move the mouse to the new position, and release the button.

- If that should not be possible, the boolean attribute `noresize` in the `frame` element can be specified.

  In HTML, one writes `<frame ... noresize>`. In XHTML, one must write `<frame ... noresize="noresize" />`.

Frame Definition (9)

- Normally, the browser uses automatically scrollbars, if the contents does not fit into the frame.
- With `scrolling=no` the scrollbars can be switched off, with `scrolling=yes` they are activated even if the contents does fit.

  The default is `scrolling=auto`.
- Switching off scrollbars is bad style. How should the user see the rest of the frame contents?

Frame Definition (10)

- Normally, the browser displays delimiting lines between the frames.
- These can be switched off with `frameborder=0`.

  `frameborder` has only the possible values `0` and `1`. One cannot make the border thicker. `frameborder=0` must be set in both frames that share the border.

- With `marginwidth` and `marginheight` one can request empty space between the frame borders and the frame contents.

  `marginwidth` sets the empty space on the left and te right hand side, and `marginheight` sets the empty space above and below the frame contents.

Frame Definition (11)

- `title` can contain a short description of the frame, and `longdesc` can point to a loger description.
- This is especially important if `src` points directly to an image file.

  In contrast to the `img`-element, there is otherwise no alternative text for people who cannot see.

Frame Definition (12)

- Declaration of the element type `frameset`:

```
<!ELEMENT frameset     - -
          ((frameset|frame)+ & noframes?)>
<!ATTLIST frameset
          %coreattrs;
          rows         %MultiLengths; #IMPLIED
          cols         %MultiLengths; #IMPLIED
          onload       %Script;       #IMPLIED
          onunload     %Script;       #IMPLIED >
```

- Declaration of the element type `noframes`:

```
<!ELEMENT noframes - - (body) -(noframes)>
<!ATTLIST noframes %attrs; >
```

Frame Definition (13)

- Declaration of the element type `frame`:

```
<!ELEMENT frame - O     EMPTY>
<!ATTLIST frame
          %coreattrs;
          longdesc      %URI;            #IMPLIED
          name          CDATA            #IMPLIED
          src           %URI;            #IMPLIED
          frameborder   (1|0)            1
          marginwidth   %Pixels;         #IMPLIED
          marginheight  %Pixels;         #IMPLIED
          noresize      (noresize)       #IMPLIED
          scrolling     (yes|no|auto)    auto >
```

Changing Frame Contents (1)

- Frames can be given names with the attribute `name` of the `frame` element type.
- In links (e.g. `a`-elements), one can use the attribute `target` to specify the frame in which the document will be loaded when the user activates the link (clicks on it).

  The attribute `target` contains the name of the frame. Besides `a`, also the following elements types have this attribue: `base`, `link`, `area`, `form`.
- If there is a frame with this name, the document is opened in this frame.

Changing Frame Contents (2)

- If there is no frame with the name specified in the `target`-attribute, a new window will be opened and the window is assigned this name.
- There are also the following special frame names:
  - ◇ `_blank`: This opens always a new window.
  - ◇ `_self`: The frame that contains the link.
  - ◇ `_parent`: The frameset that contains this frame (removes last step of the division into frames).
  - ◇ `_top`: Main window, removes all frames.

Changing Frame Contents (3)

- By using `_blank` or new frame names, one can open new browser windows.
- Thus, frames not only refer to sections in a single window, but one can control the contents of several windows with frames.

  Some users consider it impolite to open new windows on their desktop. Also, this means that the "Back" button will not work.
- The HTML 4.01 Strict DTD does not contain the attribute `target`.

  I do not understand the reason. It is not declared as "deprecated".

Inline Frames: `iframe`

- With the element type `iframe`, one can insert a frame like a picture into a web page.

  As in frames, one refers with the attribute `src` to the frame contents. It also has the attributes `longdesc`, `name`, `frameborder`, `marginwidth`, `marginheight`, `scrolling` (see `frame`). With the attributes `width` and `height`, one can define the size of the frame.

- `iframe` is an inline element.
- It is only contained in the HTML 4.01 Transitional DTD. The same effect can be reached with `object`.