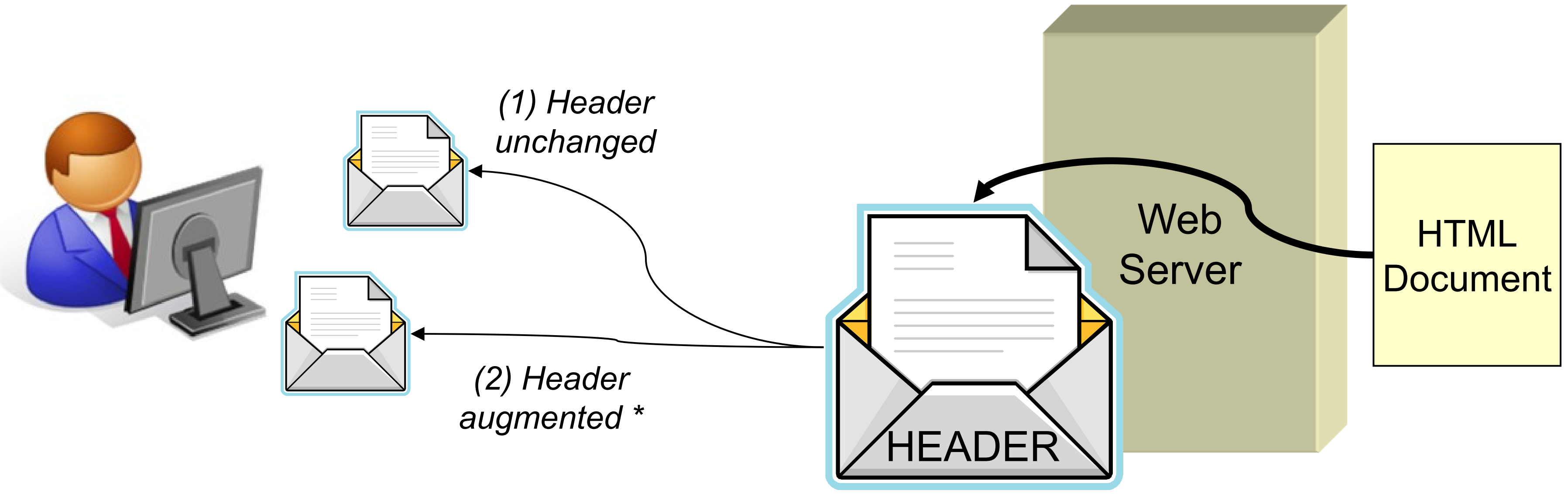
Character Encoding

- ASCII (American Standard for Information Interchange)
  - 7-bit Code = 128 characters
  - no special characters
- ISO 8859-1
  - Latin Alphabet encoding (256 characters)
  - includes special characters
- Windows-1252
  - based on ISO 8859-1
  - no control characters in 0x80 to 0x9F range
- Unicode
  - more than 100.000 characters supported
  - first 128 characters = ASCII, first 256 characters = ISO 8859-1
  - Most important charsets: UTF-8 and UTF-16

Defining Character Encoding



- Web Server can define default encoding (e.g., httpd.conf – addDefaultCharset)
- HTML/XML document may specify encoding and thereby modify document header

* no encoding specified by server

Specifying Character Encoding

- XML

```
<?xml version="1.0" encoding="utf-8"?>
```

- HTML

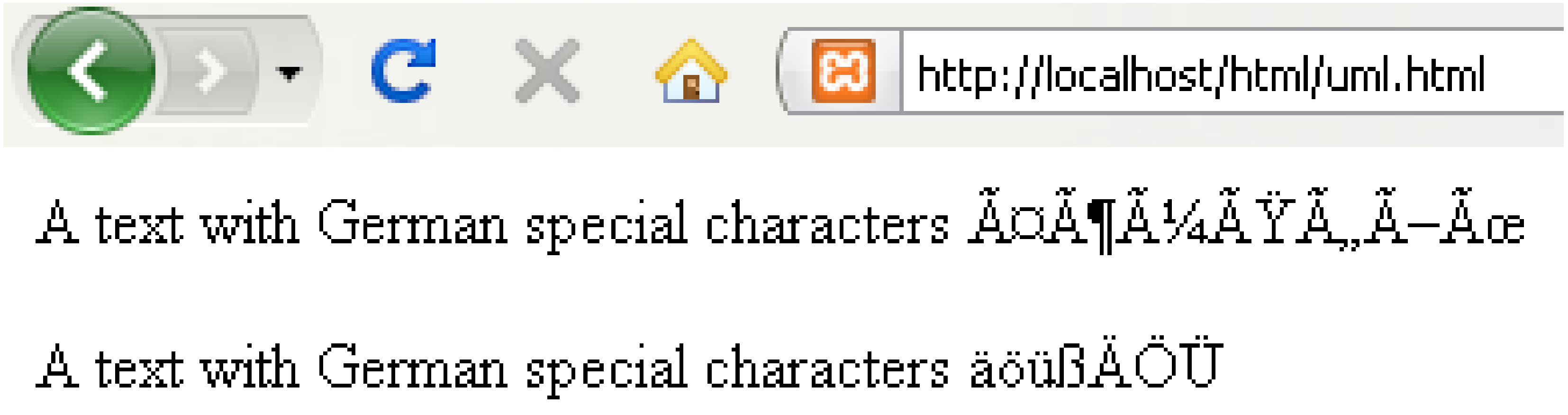
```
<meta http-equiv="Content-Type"
  content="text/html;charset=ascii" >
```

- If both are specified, XML spec has priority

- Encoding can only be set if no encoding is "enforced" by server
- This default encoding can only be changed by the admin or a server-side script (e.g., PHP, ASP.NET)

Encoding Examples: ASCII

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=ascii" >
</head>
<body>
<p> A text with German special characters äöüßÄÖÜ </p>
<p> A text with German special characters &auml;&ouml;&uuml;
&szlig;&Auml;&Ouml;&Uuml; </p>
</body>
</html>
```

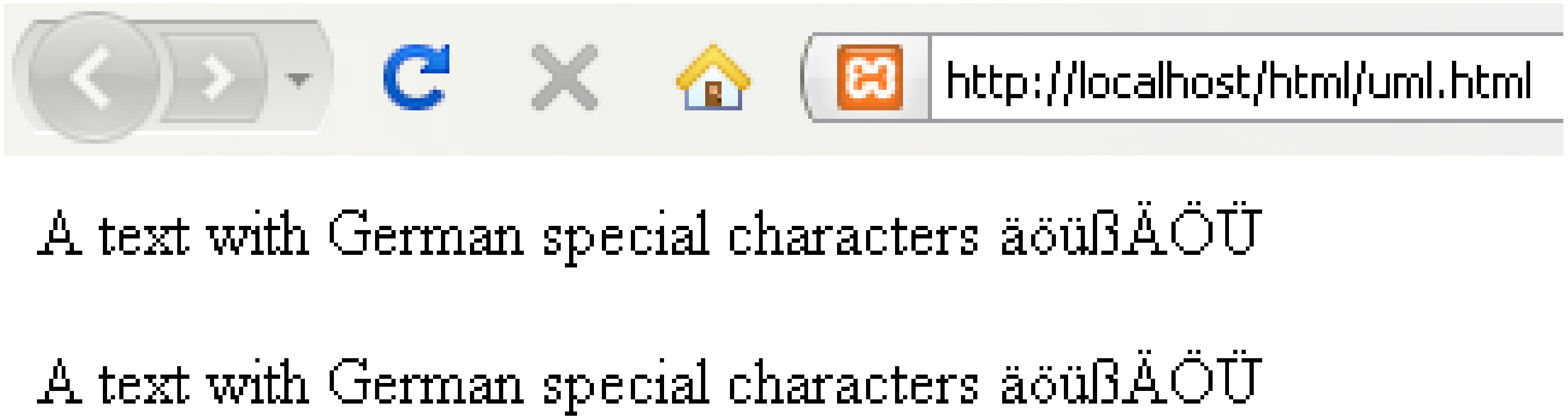


A text with German special characters Ã¤Ã¶Ã¼ÃŸÃ„Ã–Ãœ

A text with German special characters äöüßÄÖÜ

Encoding Examples: utf-8

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
</head>
<body>
<p> A text with German special characters äöüßÄÖÜ </p>
<p> A text with German special characters &auml;&ouml;&uuml;
&szlig;&Auml;&Ouml;&Uuml; </p>
</body>
</html>
```

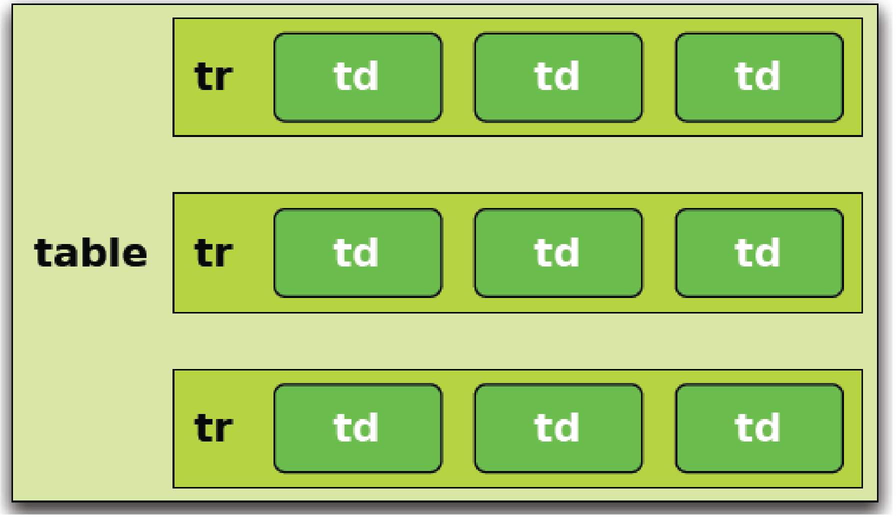

http://localhost/html/uml.html

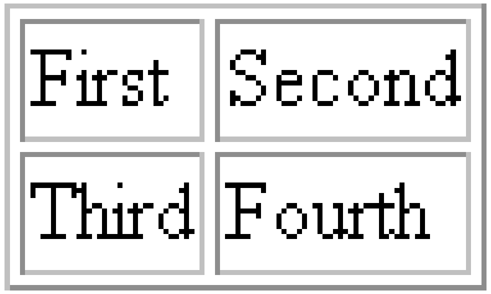
A text with German special characters äöüßÄÖÜ

A text with German special characters äöüßÄÖÜ

Tables

```
<table border="1">
  <tr>
   <td>First</td>
   <td>Second</td>
  </tr>
  <tr>
   <td>Third</td>
   <td>Fourth</td>
  </tr>
</table>
```





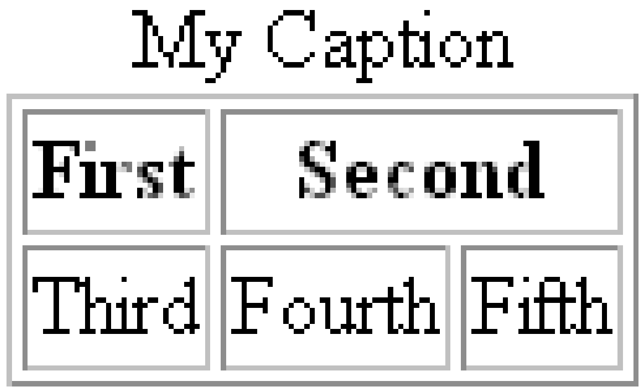
First	Second
Third	Fourth

Table with Headers

```
<table border="1">
<caption>My Caption</caption>
<thead>
  <tr>
   <th>First</th>
   <th colspan="2">Second</th>
  </tr>
 </thead>
<tbody>
  <tr>
   <td>Third</td>
   <td>Fourth</td>
   <td>Fifth</td>
  </tr>
</tbody>
</table>
```

Table Caption

Cell spanning 2 columns



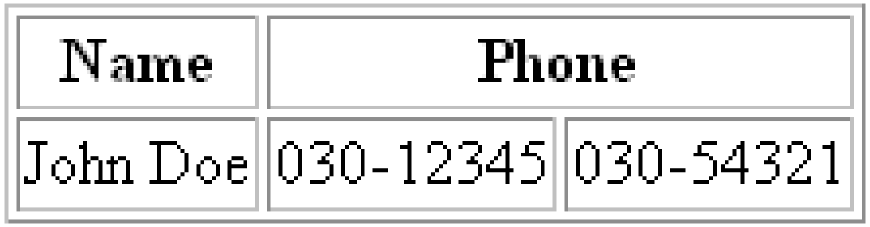
My Caption

First	Second	
Third	Fourth	Fifth

Table Cell Spanning: Columns

- **Cell spanning two colums:**

```
<table border="1">
<tr>
  <th>Name</th>
  <th
   colspan="2">Phone</th>
</tr>
<tr>
  <td>John Doe</td>
  <td>030-12345</td>
  <td>030-54321</td>
</tr>
</table>
```

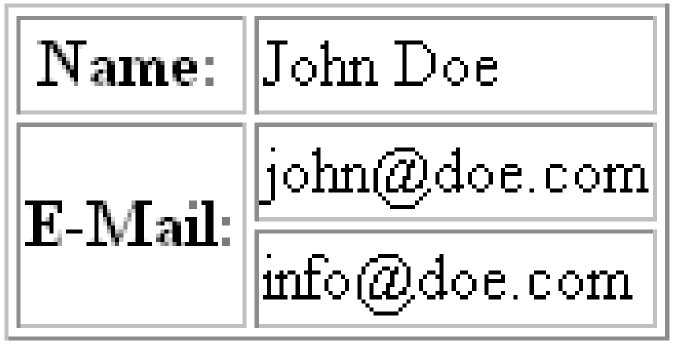


Name	Phone	
John Doe	030-12345	030-54321

Table Cell Spanning: Rows

- **Cell spanning two rows:**

```
<table border="1">
<tr> <th>Name:</th>
  <td>John Doe</td>
</tr>
<tr>
  <th rowspan="2">E-Mail:</th>
  <td>john@doe.com</td>
</tr>
<tr>
  <td>info@doe.com</td>
</tr>
</table>
```



Name:	John Doe
E-Mail:	john@doe.com
	info@doe.com

Table Width

- Relative and absolute width can be specified
- Overflow with absolute width specification results in the display of scrollbars

```
<table border="1" width="50%">
```

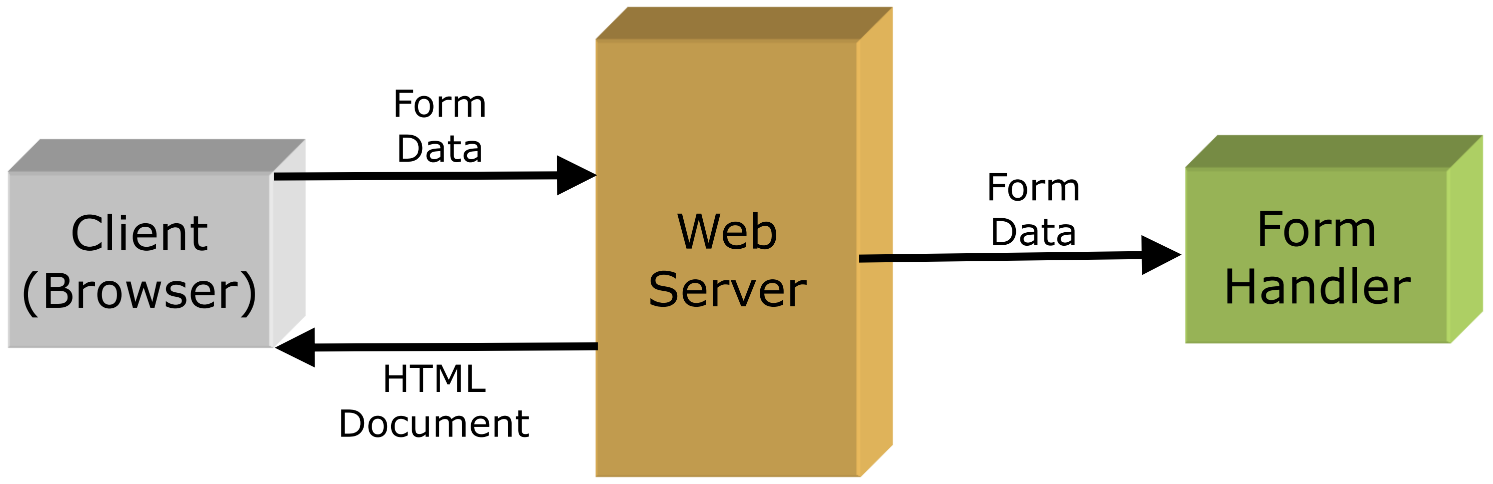
- Table occupies 50 per cent of the users screen width

- Absolute width in pixels:

```
<table border="1" width="350px">
```

Exercises 3.1 + 3.2

Forms



- Web Server sends HTML document (containing the form) to the client
- User enters data and submitts form
- Form data is passed to a form handler, which stores/processes the data

Forms

- Form data is either passed via HTTP GET (Arguments are transferred in the URL) or HTTP POST (Arguments are transferred transparently in the body of the HTTP request)

- **The Form Tag**

```
<form id="myform" action="form_handler_URL"
  method="get|post">

<!-- form elements -->

</form>
```

Form Elements

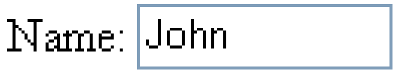
Input Boxes

- All form elements have to have a name attribute (used as "variable name" for form data)
- All form elements should have an ID value (identification of form elements)

- **Text Input Boxes**

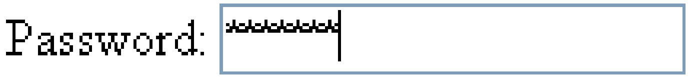
```
<input type="text" name="MyText" id="idMyText" value="initial
   value" size="size_of_field"
   maxlength="max_characters"_allowed" />
```

- Example:


Name: John

```
<p>Name: <input type="text" name="StudentID" id="ID_StudendID"
   value="" size="10" maxlength="10" /></p>
```

- **Password Input Boxes**


Password: *******

```
<p>Password: <input type="password" name="Password"
   id="ID_Password" value="" size="20" maxlength="40" /></p>
```

Form Elements

Labels & Radio Buttons

- Labels

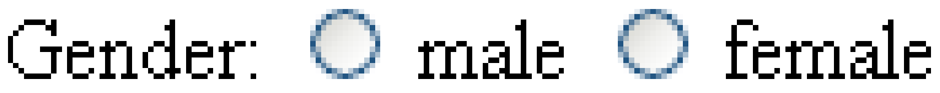
```
<label for="id_of_related_tag">Label Text</label>
```

- Radio Buttons

```
<input type="radio" name="control_group_name"
   id="ID_control" checked="checked"
   value="value_if_selected" />
```

- Example:

```
<p>Gender:
<input type="radio" name="gender" id="ID_male" value="male"
   /> male
<input type="radio" name="gender" id="ID_female"
   value="female"/> female </p>
```



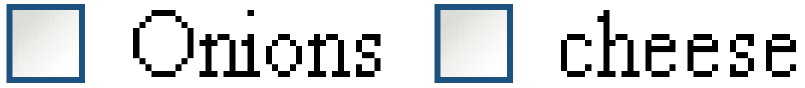
Form Elements

Checkboxes

```
<input type="checkbox" name="field_name"
   id="id_of_checkbox" checked="checked"
   value="value_if_checked" />
```

- Example:

```
<p>
<input type="checkbox" name="sandwich_choices"
   id="id_onions" value="onions" />  Onions
<input type="checkbox" name="sandwich_choices"
   id="id_cheese" value="cheese" />  cheese
</p>
```

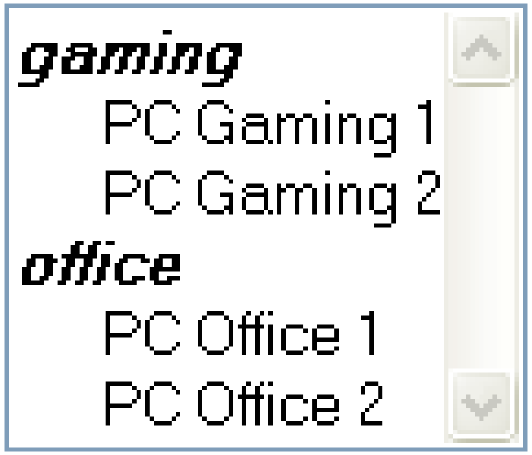


Form Elements

Listboxes

```
<select name="name_of_selectbox" id="id_value"
   size="number_of_items_visible" multiple="multiple">


<optgroup label="gaming">
   <option>PC Gaming 1 </option>
   <option>PC Gaming 2 </option>
</optgroup>
<optgroup label="office">
   <option>PC Office 1 </option>
   <option>PC Office 2 </option>
</optgroup>
</select>
```



Form Elements

Large Text Areas

- contains up to 1024 characters
- text can contain line breaks

```
<textarea name="..." cols="number_of_columns"
  rows="number_of_rows">default_value</textarea>
```

- Example:

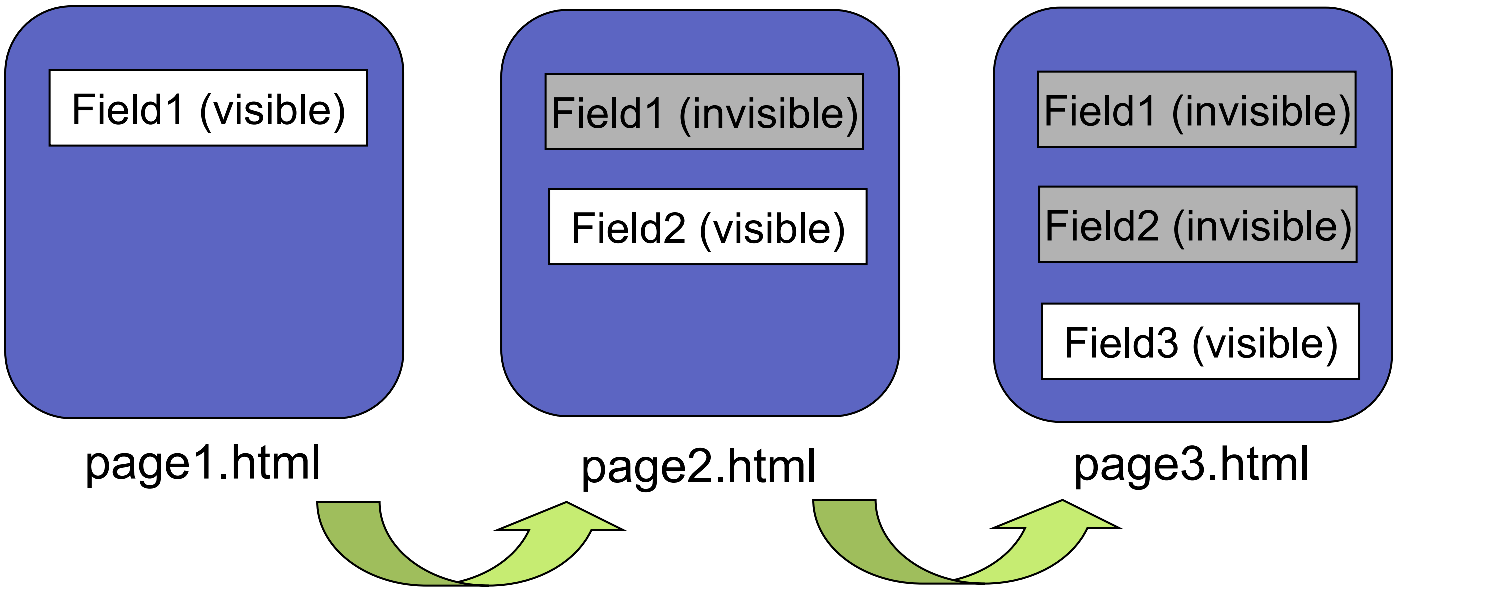
```
<textarea name="text1" cols="50" rows="10">Lorem ipsum
  dolor sit amet, consectetuer adipiscing elit.
  Maecenas luctus lectus ut pede. Donec aliquet mauris
  non mi. Nam placerat dolor ultricies arcu. Lorem
  ipsum dolor sit amet, consectetuer adipiscing elit.
  Nunc faucibus tincidunt ipsum. Sed pulvinar mi vitae
  neque. Sed fringilla, nibh sed interdum porta, mauris
  nisl iaculis odio, sed gravida mauris ligula vel
  pede. Etiam congue tincidunt mi. Nullam sagittis
  libero mollis felis. Nulla arcu. Cras libero orci,
  pretium eu, tempor ac, tristique et, nisl. Proin
  fermentum, enim sed pellentesque tincidunt, ipsum
  dolor dictum elit, eget varius velit massa non elit.
  Proin id est. Duis et turpis condimentum enim
  lobortis gravida. Etiam tempus dictum pede. Nunc
  vehicula lectus vel erat. </textarea>
```

Form Elements

Hidden Fields

- are invisible
- are used to store data (across pages)

```
<input type="hidden" name="field_name"
   value="field_value" />
```



Form Elements

Buttons and Images

- Buttons

```
<input type="button" name="button_name"
 value="button_text" />
```

- Images
  - can be used as "graphical buttons"
  - make only use within a form if tied to a JavaScript event handler

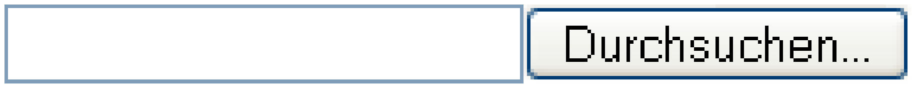
```
<input type="image" name="field_name" src="url_to_image" />
```

Form Elements

File Fields

- allows to attach file data to submitted form data
- to be used within a form the form has to
  - set encoding to multipart (see below)
  - use POST to deliver form data

```
<input type="file" name="field_name"
  size="displayd_size" />
```



- form tag has to look like this:

```
<form action="handler" method="post"
  enctype="form/multipart">
```

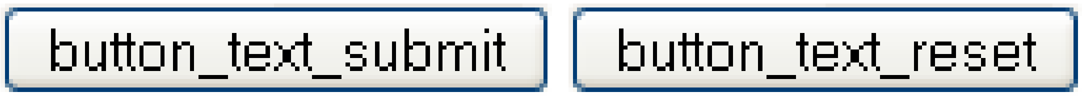
Form Elements

Submit and Reset Buttons

- SUBMIT sends the form data to the form handler
- RESET causes to form to re-load its default values

- `<input type="submit" name="submit" id="submit" value="button_text_submit"/>`

- `<input type="reset" name="reset" id="reset" value="button_text_reset"/>`

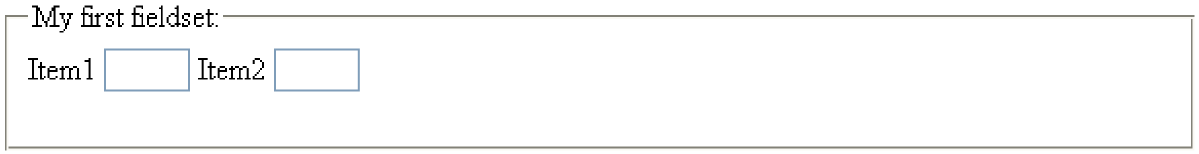


Form Elements

Grouping and Accessibility

- Grouping of form elements via a fieldset

```
<fieldset>
  <legend>Caption for grouped fields </legend>
  <input ... />
  <input ... />
</fieldset>
```



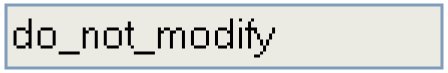
- Form Field Accessibility
  - Tabindex specified in which order fields are focussed after using the tab key
  - accesskey provides a shortcut via using <ALT>+<accesskey>

```
<fieldset>
  <legend>Caption for grouped fields </legend>
  <input ...  tabindex="1" accesskey="F"/>
  <input ...  tabindex="2" accesskey="E"/>
</fieldset>
```

Form Elements

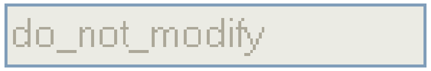
Preventing Change of Data

- In some cases, users should not be allowed to modify the contents of a form element
- Method 1: setting a field to read-only


do_not_modify

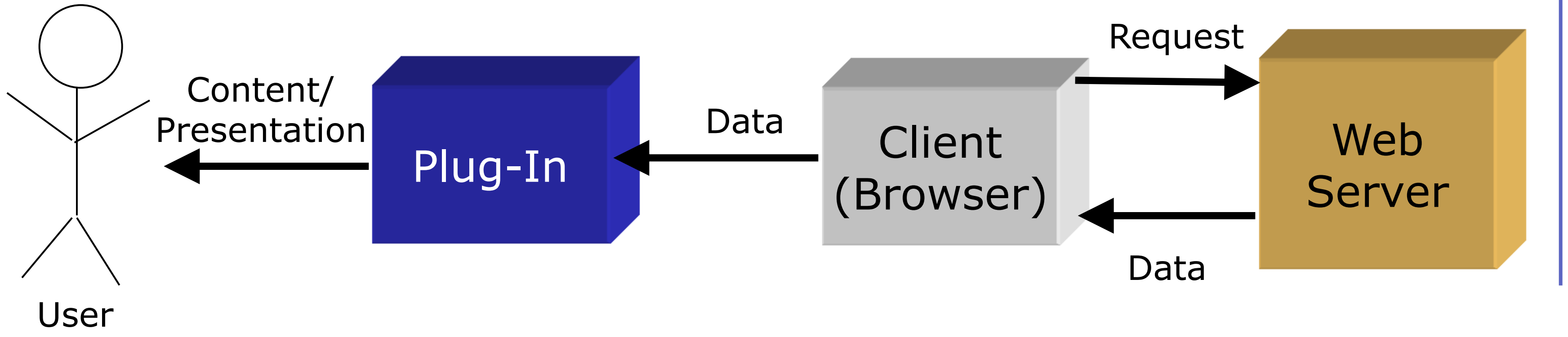
```
<input type="text" name="t1" id="id_t2"
  value="do_not_modify" readonly="readonly" />
```

- Method 2: disabling a field


do_not_modify

```
<input type="text" name="t2" id="id_t2"
  value="do_not_modify" disabled="disabled" />
```

Plug-Ins



- Plug-Ins augment the Browser's capability to display Non-HTML contents such as Java Applets, flash videos, MIDI files, ...
- Objects represent non-HTML contents embedded to an HTML page
- <object> tag replaces deprecated <embed> and <applet> tag

Embedding Non-HTML Content

- **<embed> (deprecated)**
  - Embed tag automatically determines for registered content handlers (flash videos, MIDI files, WAV files, MP3 files, ...)
  - Problem: if content handler (e.g., Apple Quicktime video plug-in) is not installed, nothing is displayed (some browsers provide "install missing plugins" functionality)
- Syntax: <embed src="URL_or_Path" </embed>
- Example (MIDI):

```
<embed src="c:\temp\test.mid">
</embed>
```

- Example (flash video):

```
<embed src="http://www.youtube.com/v/lJwgP44Ap9E">
</embed>
```

The <object> Tag

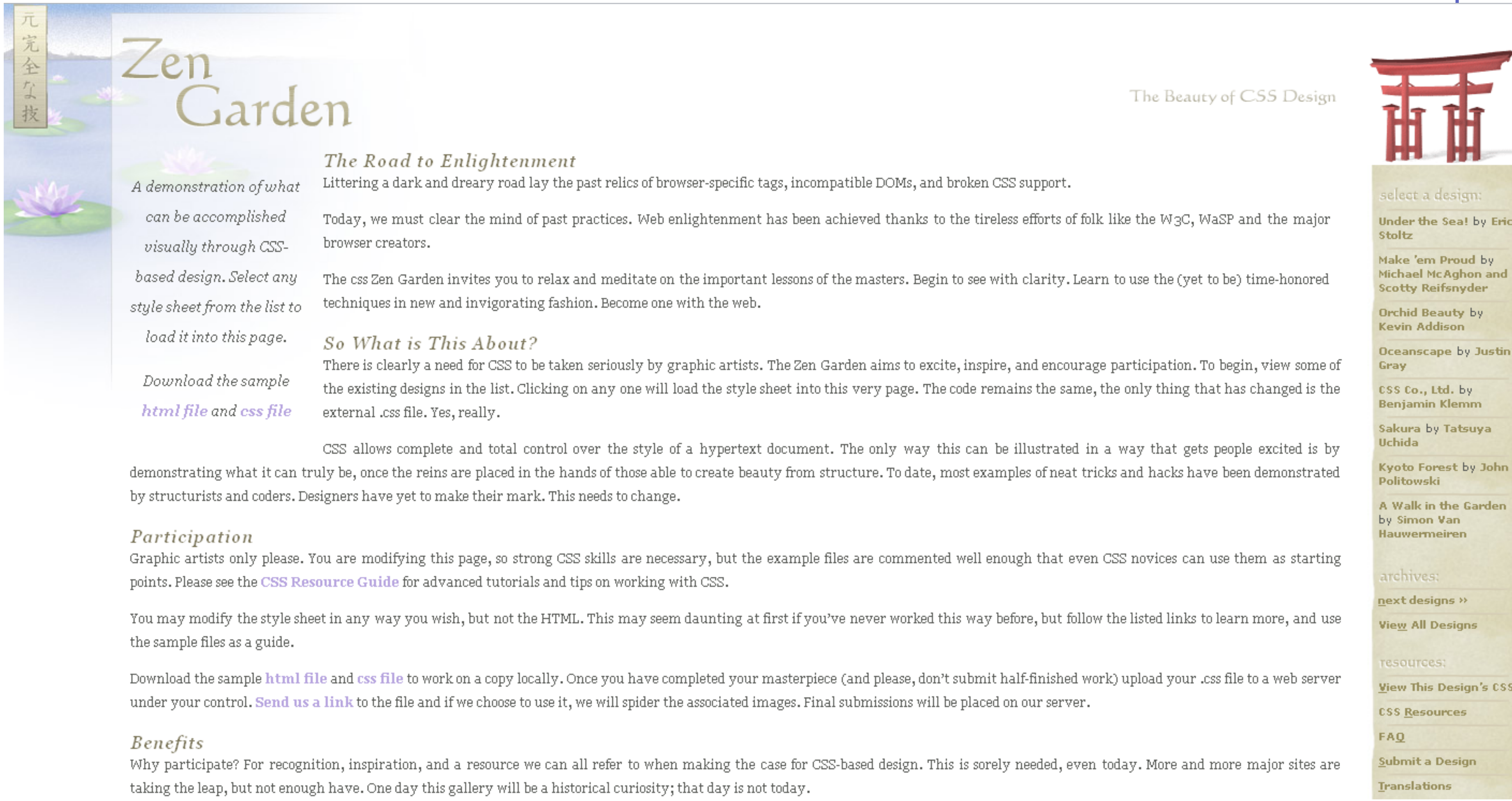
```
<object classid="..." id="ID_value"
   codebase="base_for_object_code_URL" codetype="MIME_type">
  <param name="Parameter_name" value="parameter_value" />
  ...
</object>
```

- **Problem:** some new browsers do not notice the <embed> tag while older browsers may not notice the <object> tag.
- **Solution:** combination of both
- **Description:** newer browsers interpret <embed> tag within an <object> tag as a false parameter, older browser only interpred the <embed> tag
- Example (youtube.com-generated code):

```
<object width="425" height="355">
   <param name="movie"
   value="http://www.youtube.com/v/lJwgP44Ap9E"></param>
   <param name="wmode" value="transparent"></param>
   <embed src="http://www.youtube.com/v/lJwgP44Ap9E"
   type="application/x- shockwave-flash"
   wmode="transparent" width="425" height="355"></embed>
</object>
```

Exercises (3.3, 3.4),3.5

CSS



Zen Garden

The Beauty of CSS Design

A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.

Download the sample html file and css file

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP and the major browser creators.

The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.

So What is This About?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by demonstrating what it can truly be, once the reins are placed in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks have been demonstrated by structurists and coders. Designers have yet to make their mark. This needs to change.

Participation

Graphic artists only please. You are modifying this page, so strong CSS skills are necessary, but the example files are commented well enough that even CSS novices can use them as starting points. Please see the CSS Resource Guide for advanced tutorials and tips on working with CSS.

You may modify the style sheet in any way you wish, but not the HTML. This may seem daunting at first if you've never worked this way before, but follow the listed links to learn more, and use the sample files as a guide.

Download the sample html file and css file to work on a copy locally. Once you have completed your masterpiece (and please, don't submit half-finished work) upload your .css file to a web server under your control. Send us a link to the file and if we choose to use it, we will spider the associated images. Final submissions will be placed on our server.

Benefits

Why participate? For recognition, inspiration, and a resource we can all refer to when making the case for CSS-based design. This is sorely needed, even today. More and more major sites are taking the leap, but not enough have. One day this gallery will be a historical curiosity; that day is not today.

select a design:

- Under the Sea! by Eric Stoltz
- Make 'em Proud by Michael McAghon and Scotty Reifsnyder
- Orchid Beauty by Kevin Addison
- Oceanscape by Justin Gray
- CSS Co., Ltd. by Benjamin Klemm
- Sakura by Tatsuya Uchida
- Kyoto Forest by John Politowski
- A Walk in the Garden by Simon Van Hauwermeiren

archives:

- next designs »
- View All Designs

resources:

- View This Design's CSS
- CSS Resources
- FAQ
- Submit a Design
- Translations

http://www.csszengarden.com/

Cascading Style Sheets (CSS)

- CSS describe the presentation of data written in a markup language (e.g., HTML)
- **Without CSS**

```
<p><b><u>Heading One</u></b></p>
<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. </p>
<p><b><u>Heading Two</u></b></p>
<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. </p>
```

- **With CSS**

```
<head>
    <style type="text/css">
    p.heading {font-weight: bold; text-decoration: underline; }
    </style>
</head>
<p class="heading">Heading One</p>
<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. </p>
<p class="heading">Heading Two</p>
<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. </p>
```

CSS Style Definition

- **Syntax:**

```
<style type="MIME_type" media="destination_media">
/* comments in a stylesheet */
</style>
```

- MIME_type is usually "text/css"
- Possible media type:
  - all - all media type devices
  - aural - speech and sound synthesizers
  - braille - braille tactile feedback devices
  - embossed - paged braille printers
  - handheld - small or handheld devices
  - print -   printers
  - projection - projected presentations, like slides
  - screen   - computer screens
  - tty - media using a fixed-pitch character grid, such as teletypes and terminals
  - tv - television-type devices

Where to put Style Definitions

- Style definition can be place with the head section of a document or in a separate file (*.CSS)
- More than one style sheet can be associated with one HTML document

- Association of a stylesheet via <link> tag:

```
<head>
  <link rel="stylesheet" type="text/css"
  href="styles.css" />
  <link rel="stylesheet" type="text/css"
  href="another.css" />
</head>
```