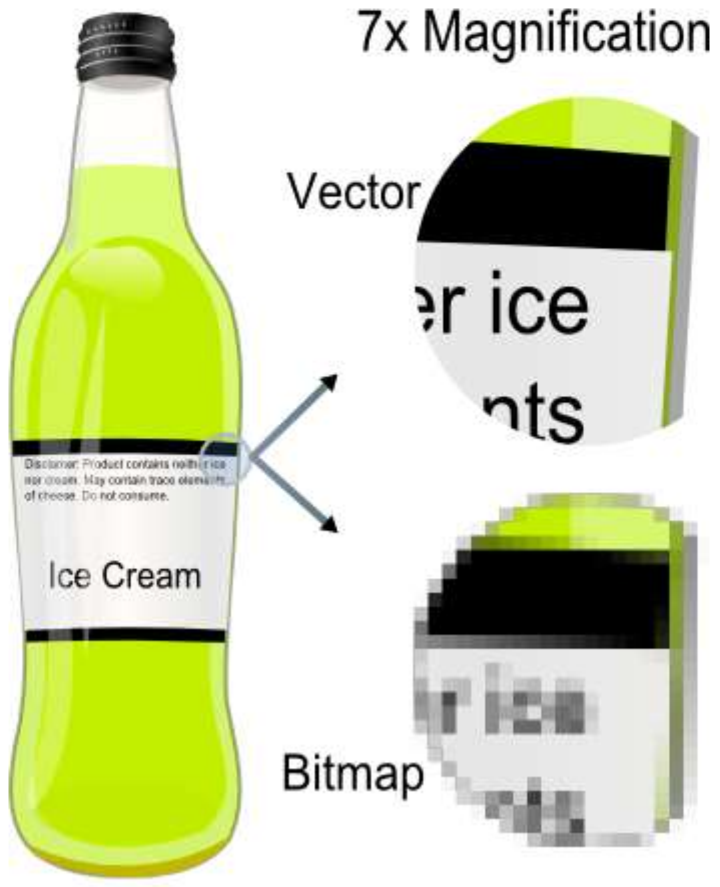
Images


7x Magnification
Vector
er ice
nts
Disclaimer: Product contains neither ice nor cream. May contain trace elements of cheese. Do not consume.
Ice Cream
Bitmap

Source: http://en.wikipedia.org/wiki/Vector_graphics

- Vector graphics are stored as a collection of vectors
- Small memory size
- Complex rendering

- Bitmap/Raster Graphics are stored as array of pixels
- Large memory size
- Simple rendering

Bitmap Graphics

- Modern graphics displays are raster based
- This just means that they display a grid of pixels, where each pixel color can be set independently
- Individual pixels are usually formed from smaller red, green, and blue (RGB) subpixels. If you look very closely at a TV screen or computer monitor, you will notice the pattern of subpixels
- Older style vector displays did not display a grid of pixels, but instead drew lines directly with an electron beam
- Can only be displayed by interpreting software/hardware

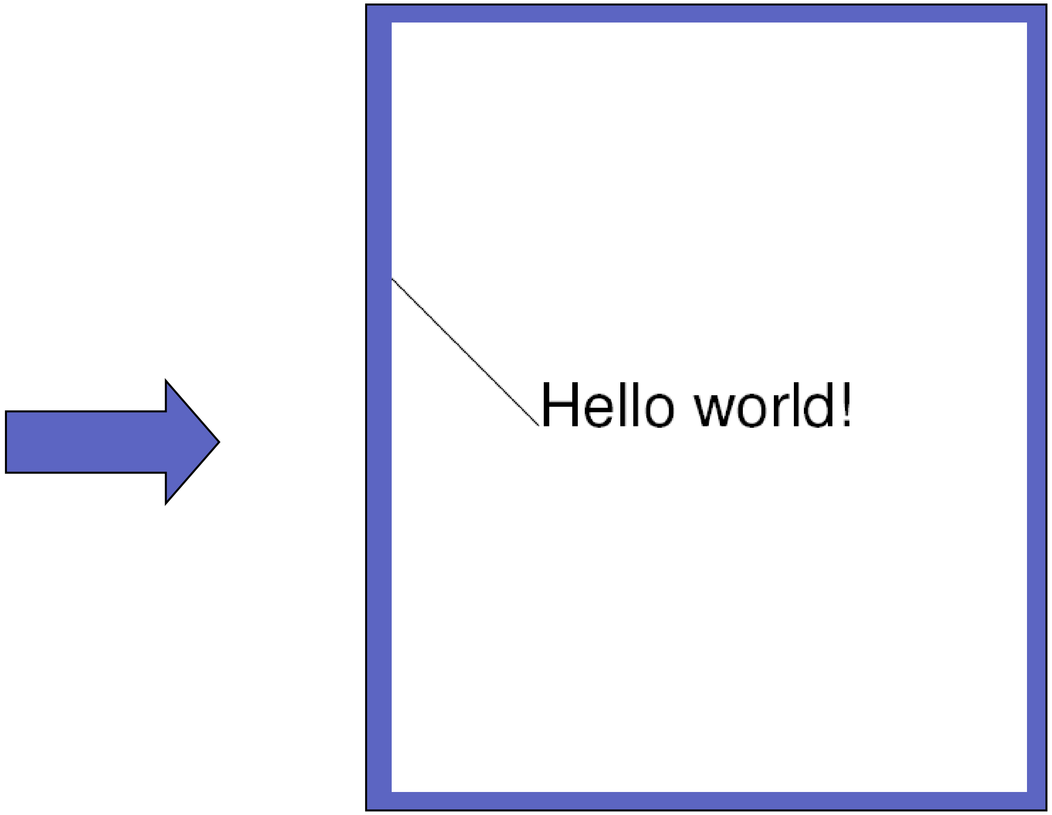
Vector Graphic Images

- A vector graphic file format is composed of analytical geometry formula representations for basic geometric shapes (e.g., line, rectangle, ellipse, etc.)
- Used primarily for storing graphics produced with technical drawing programs or "simple images"
- The graphic can be resized easily by simply changing the coefficients of the geometric
- Resizing does not effect quality

Postscript: Vector Graphics & Text

- A programming language developed in 1982 by Adobe Systems
- File Extension: *.PS
- Subset of Postscript is used by Adobe PDF
- Example:

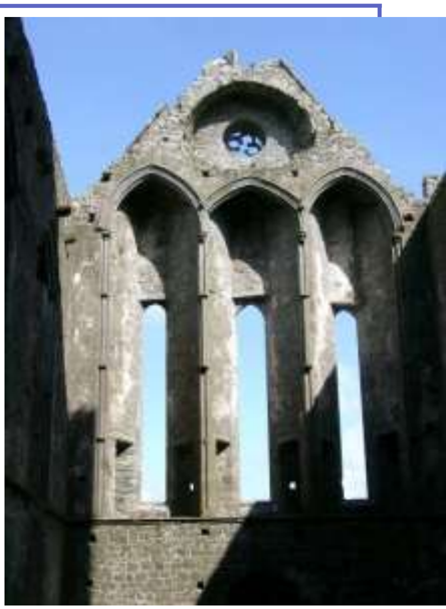
```
%!PS
/Arial findfont
40 scalefont
setfont
100 500 moveto
(Hello world!) show
100 500 moveto
0 600 lineto stroke
showpage
```

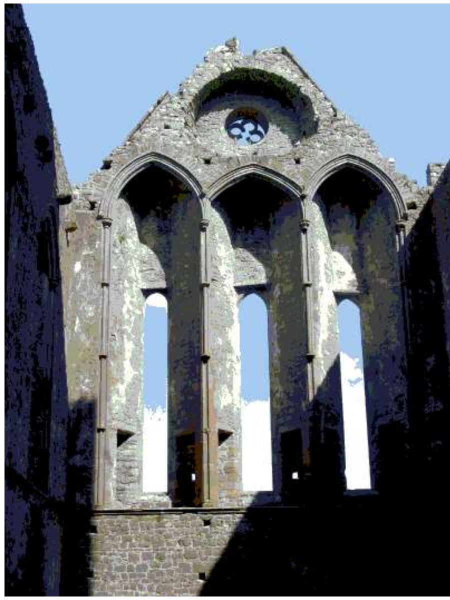


24-Bit and 8-Bit Images

- 24-Bit Bitmap Color Graphics
  - Photographic quality
  - Each pixel value is represented as three bytes (one for each primary RGB color). Thus 256 different shades of red, green and blue is possible for each pixel; 256^3 possible combined colors (16,777,216)
  - A 640 X 480 24-bit color image would require 921.6KB of storage.

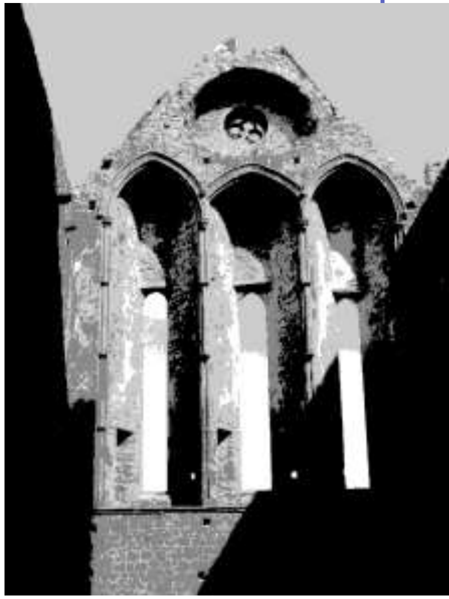
- 8-Bit Bitmap Color Graphics
  - Monitors are capable of displaying millions of colors.
  - This requires 8-bit color images have color look-up tables (CLUT), stored with them to represent which 256 colors, out of the millions possible, are to be used in the image. A 640 X 480 8-bit color image would require 307.2KB of storage (the same as 8-bit grayscale).
  - Acceptable color quality, but does not compare to 35mm photographic quality


Rock of Cashel, Ireland

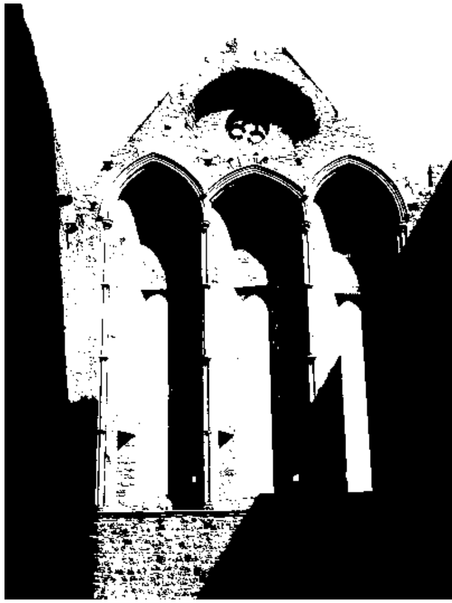


Grayscale and Monochrome Images

- A grayscale image, usually requires that each pixel be stored as a value between 0 - 255 (byte). Where the value represents the shade of gray of the pixel.
  A 640 X 480 grayscale image would require 307.2KB of storage.



- In a monochrome (black/white), image, (like the example at the right), each pixel is stored as a single 0 or 1 value (bit).
  A 640 X 480 monochrome image would require 38.4KB of storage.

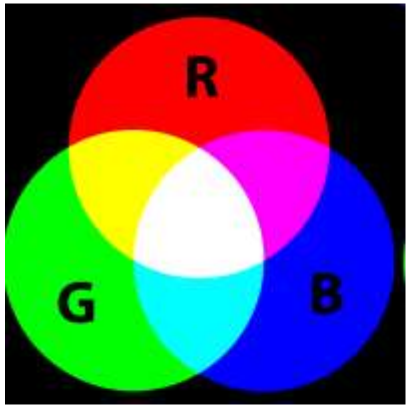


Color Models

- Define how colors are "synthesised"
- Apply to Images, Videos, Print Media, …
- Can be grouped into
  - Additive Color Models
  - Subtractive Color Models

RGB Color Model

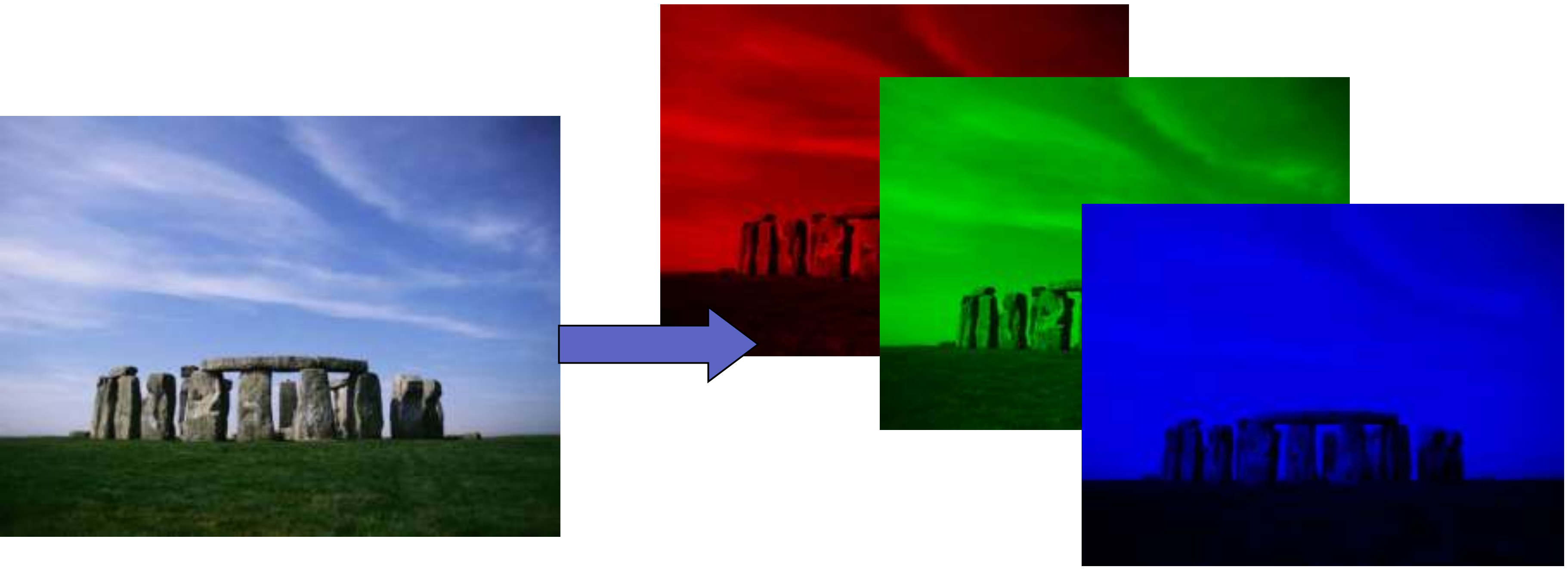
- RGB Model is an additive color model
- Colors within model: Red, Green, Blue
- Colors specified by levels of R, G, B, e.g., (255,255,255) -> WHITE, (0,0,0) -> BLACK



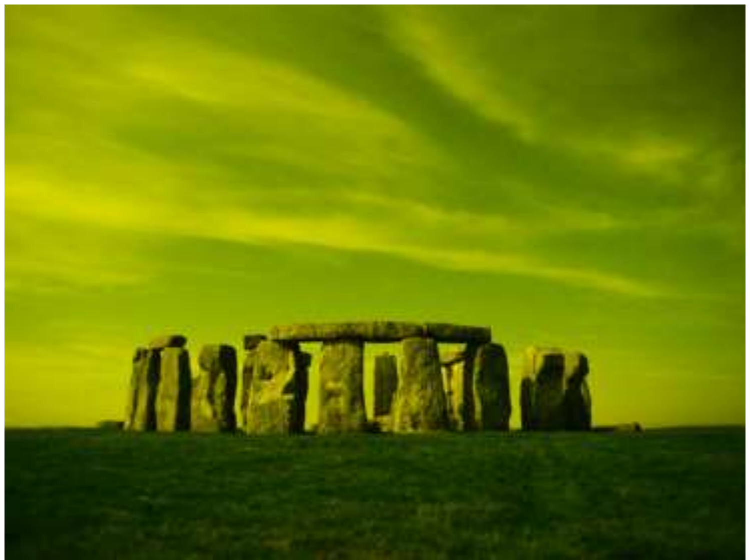
Source: en.wikipedia.org

- All colors togther spawn a Color Space
- RGB is device-independant, but synthesised color can differ across devices -> color management
- RGB is used, e.g., for computer monitors

RGB: Splitting-Up Color Channels

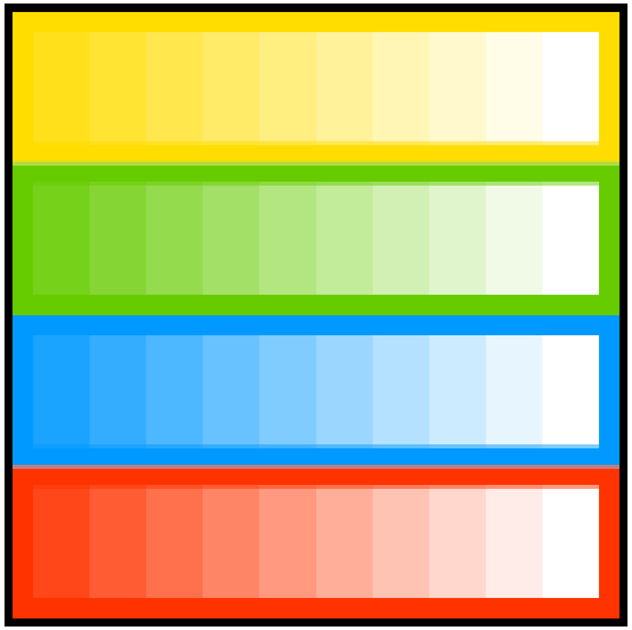


Example: R + G =



RGBA

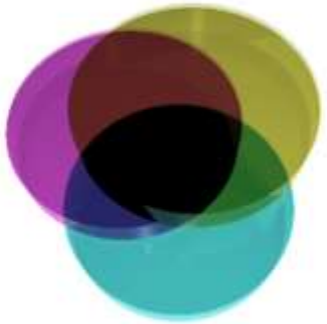
- RGBA = RGB + Alpha
- RGBA is used by CSS3
- Alpha channel specifies transparency
  - A: 0% -> fully transparent (invisible)
  - A: 100% -> fully opaque



Source: http://developer.mozilla.org/en/Canvas_tutorial/Applying_styles_and_colors

CMYK Color Model

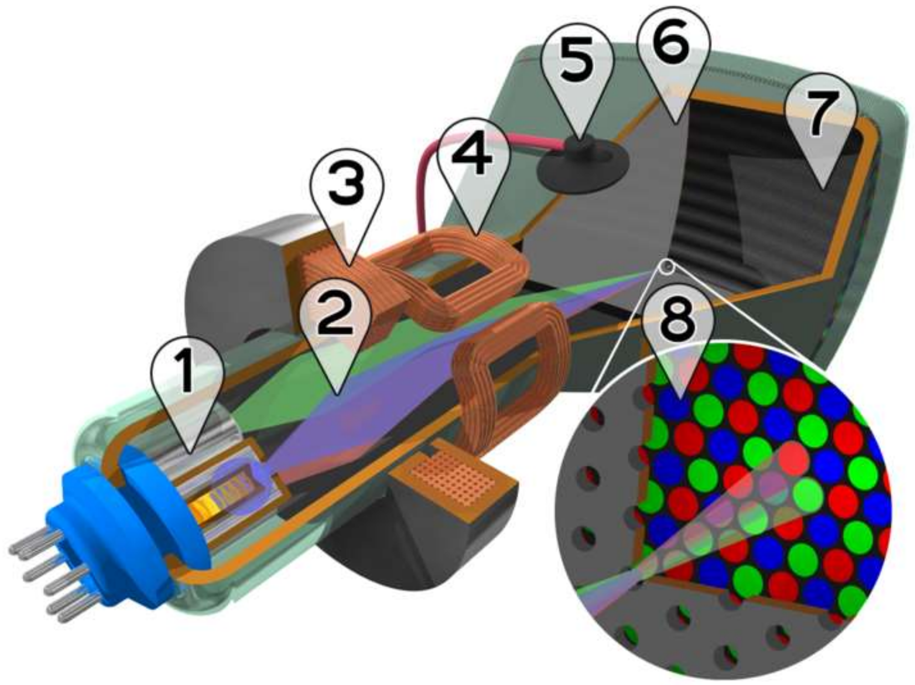
- CMYK is a subtractive color model
- Colors within model: cyan, magenta, yellow, and key (black)
- Colors specified by levels of CMYK e.g., (255,255,255,255) -> BLACK, (0,0,0,0) -> WHITE
- Conversion from RGB:
  - R = 1.0 - (C + K)
  - G = 1.0 - (M + K)
  - B = 1.0 - (Y + K)
- CMYK is used for printing (hint: compare the colors of your inkjet printer to CMYK)



Source: en.wikipedia.org

Display Technology

- CRT (cathode ray tube)
- LCD (liquid crystal display)
- TFT (thin film transistor)
- Plasma
- Film
- Print
- ...



Source: en.wikipedia.org

SVG – Scalable Vector Graphic Format

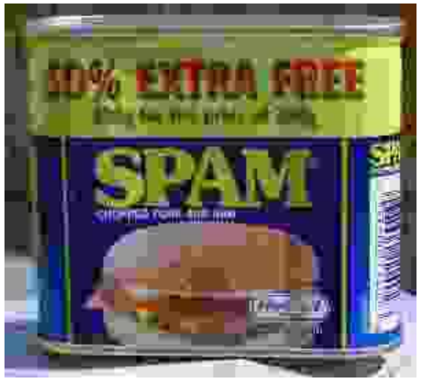
- XML-based vector format by W3C
- Rendering in Browser via Plug-In (IE) or native SVG support

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" viewBox="0 0 21000 29700"
preserveAspectRatio="xMidYMid" fill-rule="evenodd"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"><g
visibility="visible"
id="Default"><desc>Master slide</desc></g><g
visibility="visible"
id="page1"><desc>Slide</desc><g><desc>Drawing</desc>
<g><g style="stroke:none;fill:rgb(255,255,0)"><path d="M
5249,1500 L 4491,3984 2000,3984 4023,5534 3263,8000
5249,6490
7237,8000 6477,5534 8500,3984 6009,3984 5249,1500
5249,1500 Z"/></g><g style="stroke:rgb(0,0,0);fill:none">
<path style="fill:none" d="M 5249,1500 L 4491,3984
2000,3984 4023,5534 3263,8000 5249,6490 7237,8000
6477,5534 8500,
3984 6009,3984 5249,1500"/></g><g/></g></g></g></svg>
```

Compression

- Almost all graphic/video formats incorporate some variation of a *compression* technique due to the large storage size of image files. These can be classified into either **lossless** or **lossy** formats.
- Lossless formats compress all of the original captured/created data of the image/graphic using algorithms that allow the original data of the file to be recreated *without loss* of any data, hence the name.
- Lossy formats discard data when storing images. The data discarded is in most cases beyond the ability of the human vision system and thus there is no discernible difference between the original image and the compressed image.

Original Photo

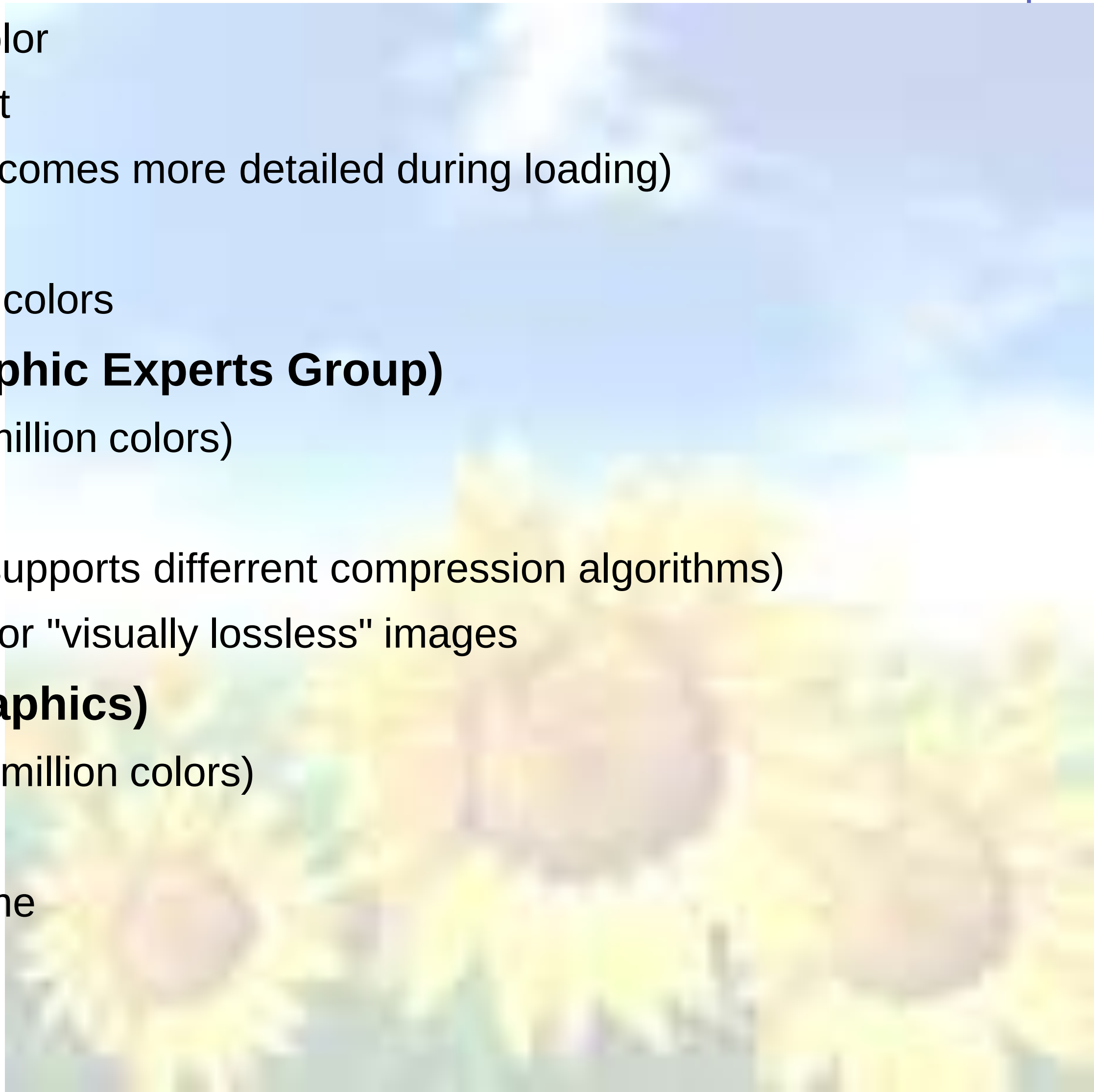


Compressed Photo

Bitmap Graphic Format

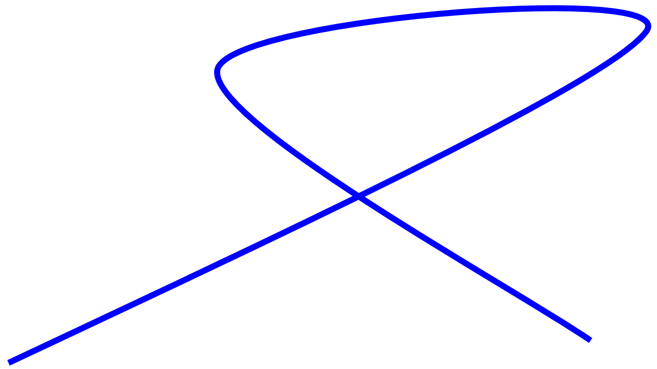
GIF, JPG, PNG

- **GIF (Grafic Interchange Format)**
  - uses Lempel-Ziv-Welch (LZW) compression (dictionary-based approach)
  - supports one transparent color
  - lossless, compressed format
  - can be interlaced (image becomes more detailed during loading)
  - supports simple animations
  - supports up to 256 different colors
- **JPG/JPEG (Joint Photographic Experts Group)**
  - supports 24-bit color (16,7 million colors)
  - no transparency support
  - lossy compression format (supports differrent compression algorithms)
  - compression ration of 1:15 for "visually lossless" images
- **PNG (Portable Network Graphics)**
  - supports 24-bit colors (16,7 million colors)
  - supports transparency
  - lossless compression scheme



Graphic Editor Software

- Vector Graphic Editors
  - Inkscape
  - OpenOffice Draw
  - Corel Draw
  - Adobe Freehand
  - ZCubes
  - …
- Bitmap Graphic Editors
  - Adobe Fireworks
  - Adobe Photoshop
  - Microsoft Paint
  - GIMP
  - Paint.NET
  - …



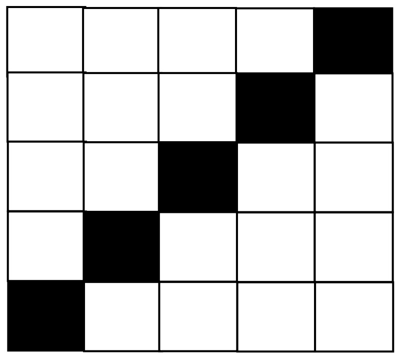
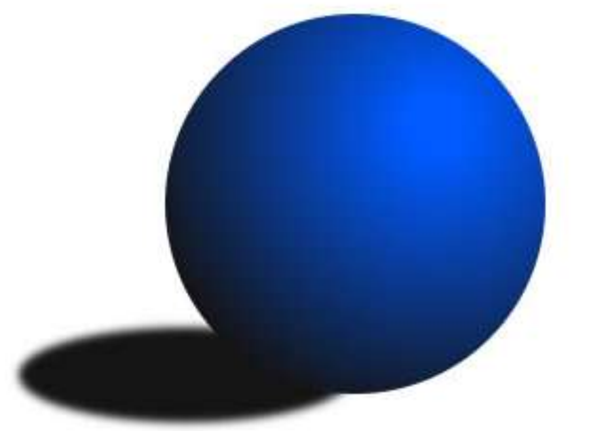
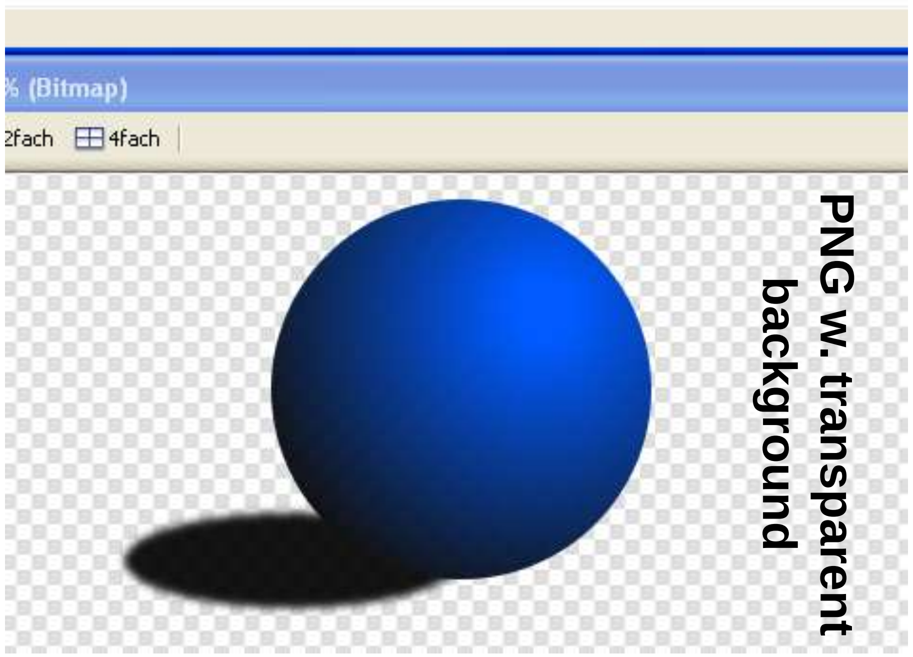


Image Transparency

- Images w/o transparency cover everything within their borders



- Transparency allows to define one/multiple colors appearing transparent during rendering



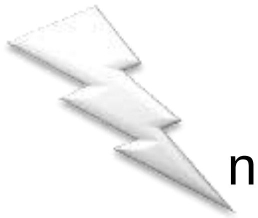
The Image Tag

- The Image Tag

```
<img src="path to image" alt="text for non-graphical user agents">
```

- Specifying image dimensions:

```
<img src="a.png" width="200" height="200">
```



not recommended
-> CSS3

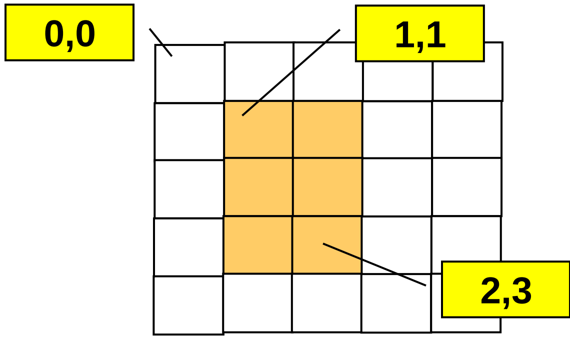
Image Maps

- Allows to specify "hotspots" for a single image file
- Coordinates are defined in pixels (x,y) from the top left corner
- First specify the image to be used as an imagemap

```
<img name="image_map" src="image_map.jpg" usemap="#m_image_map"
   alt="">
```

- Then specify map tag and define clickable regions:

```
<map name="m_image_map">
   <area shape="rect" coords="226,8,392,188"
   href="right.html" >
   <area shape="rect" coords="18,6,184,187"
    href="left.html" >
</map>
```

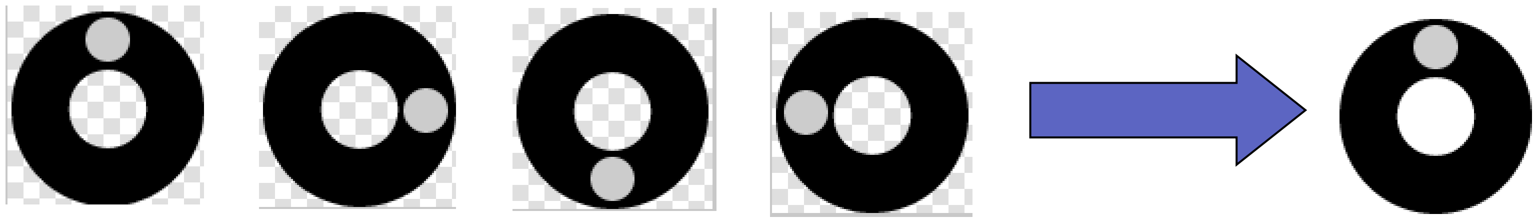


- The following shapes are supported:
  - rect: rectangular area; specified by coordinates of upper-left and lower-right corner
  - circle: circular area; specified by coordinates of center and radius
  - poly: polygon area; coordinates of each point specified

Animation

Flip-Books in the Internet Era

- Animated GIFs combine multiple images to one animation ("flip-book approach")
- Various tools are available for creating animated GIFs
- Approach:
  1. Create animation as single images
  2. Use tool to combine images
  3. Set animation properties (timing)
  4. Export to GIF

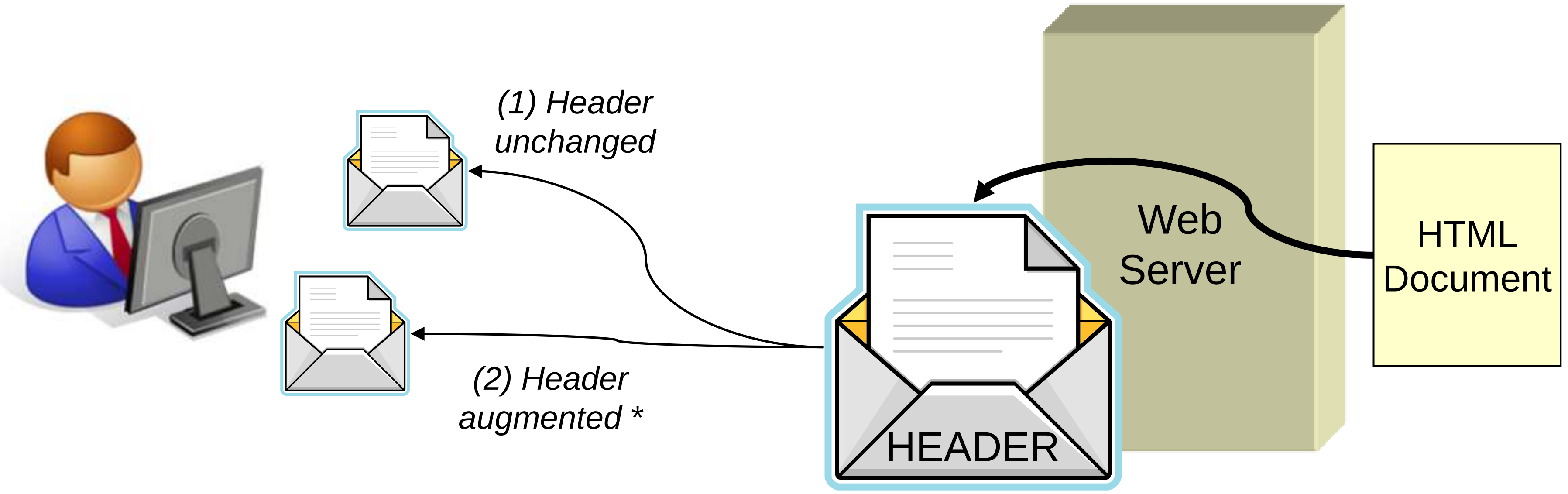


Exercises 2.6 - 2.9

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" >
```

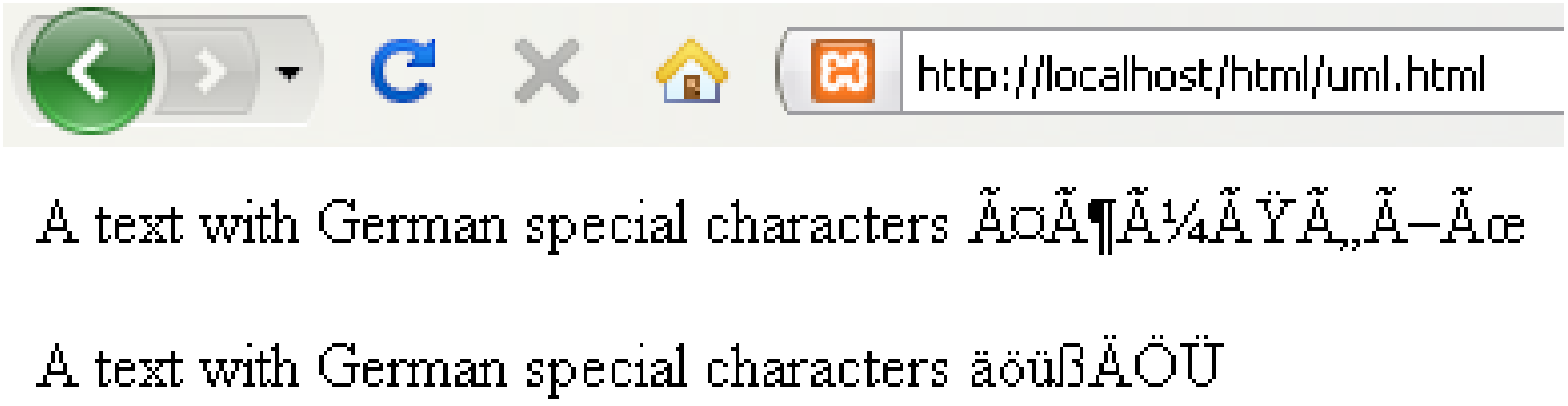
- HTML5

```
<meta charset="UTF-8">
```

- If both are specified in XHTML, 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 (XHTML): 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>
```

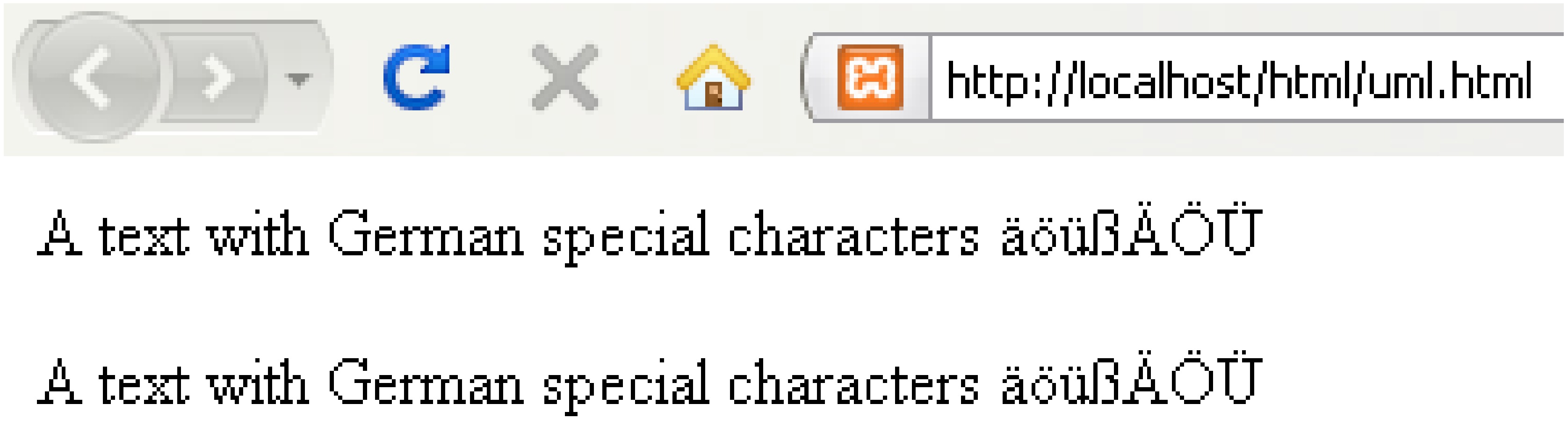


A text with German special characters Ã¤Ã¶Ã¼ÃŸÃ„Ã–Ãœ

A text with German special characters äöüßÄÖÜ

Encoding Examples (XHTML): 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>
```

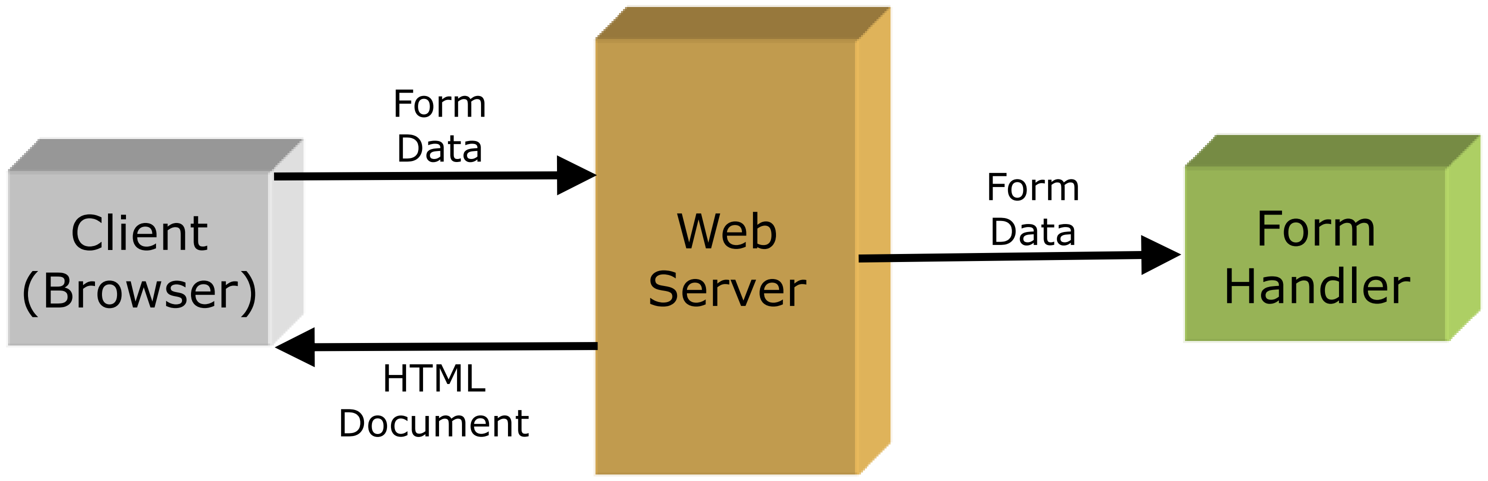


http://localhost/html/uml.html

A text with German special characters äöüßÄÖÜ

A text with German special characters äöüßÄÖÜ

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 action="form_handler_URL"
  method="get|post">

<!-- form elements -->

</form>
```

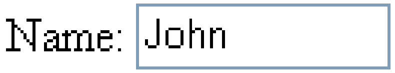
Form Elements

Input Boxes

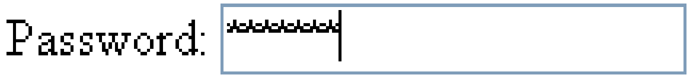
- All form elements have to have a name attribute (used as "variable name" for form data)
- All form elements should have an ID value (identification of form elements)

- **Text Input Boxes**

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

Example: 
Name: John

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


Password: *******

- **Password Input Boxes**

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

Form Elements

Labels & Radio Buttons

- Labels

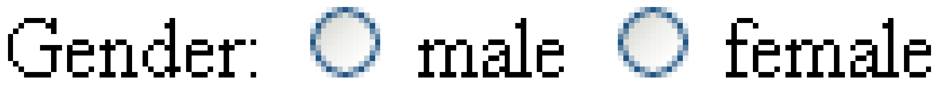
```
<label for="id_of_related_tag">Label Text</label>
```

- 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>
```



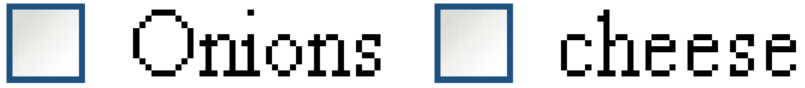
Form Elements

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>
```

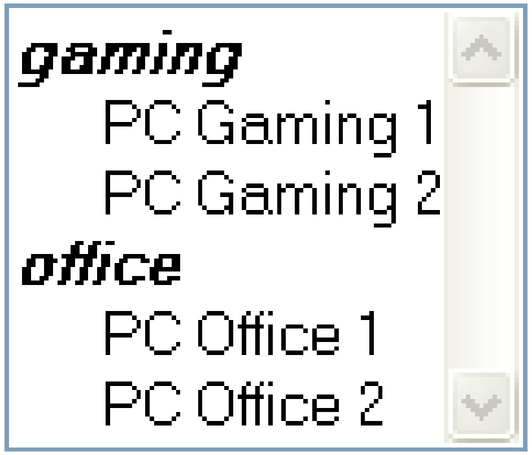


Form Elements

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:

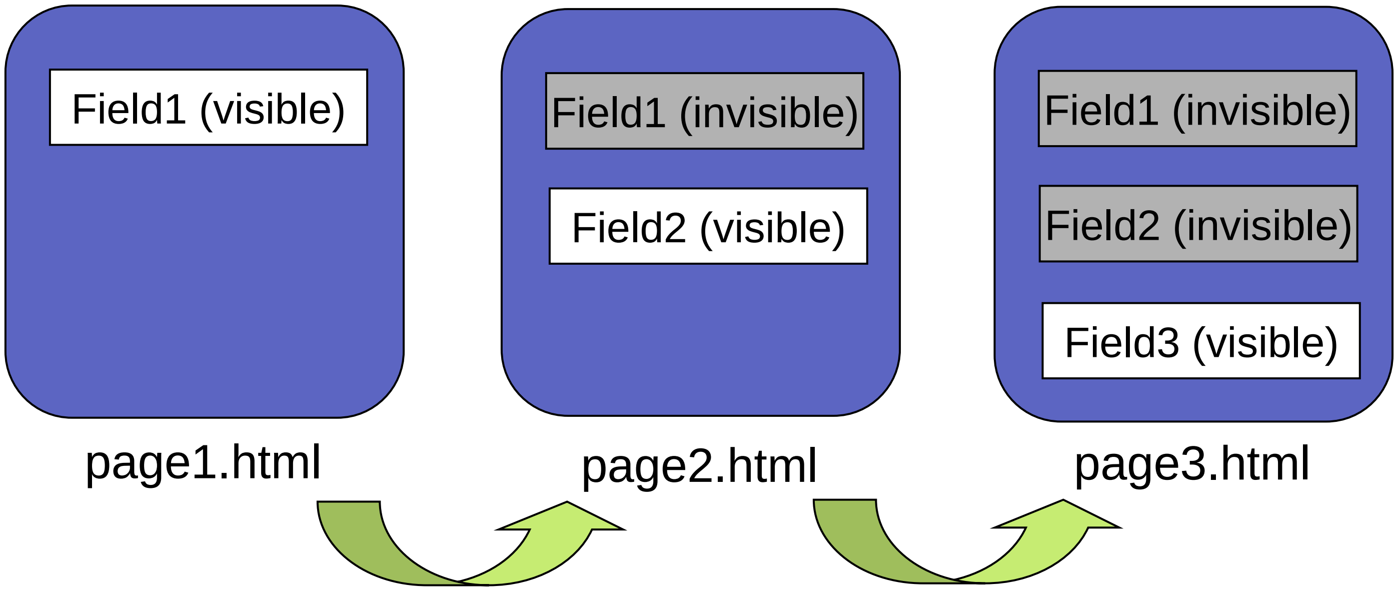
```
<textarea name="text1" cols="50" rows="10">Lorem ipsum dolor
  sit amet, consectetuer adipiscing elit. Maecenas luctus
  lectus ut pede. Donec aliquet mauris non mi. Nam placerat
  dolor ultricies arcu. Lorem ipsum dolor sit amet,
  consectetuer adipiscing elit. Nunc faucibus tincidunt
  ipsum. Sed 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">
```


Field1 (visible)
page1.html
Field1 (invisible)
Field2 (visible)
page2.html
Field1 (invisible)
Field2 (invisible)
Field3 (visible)
page3.html