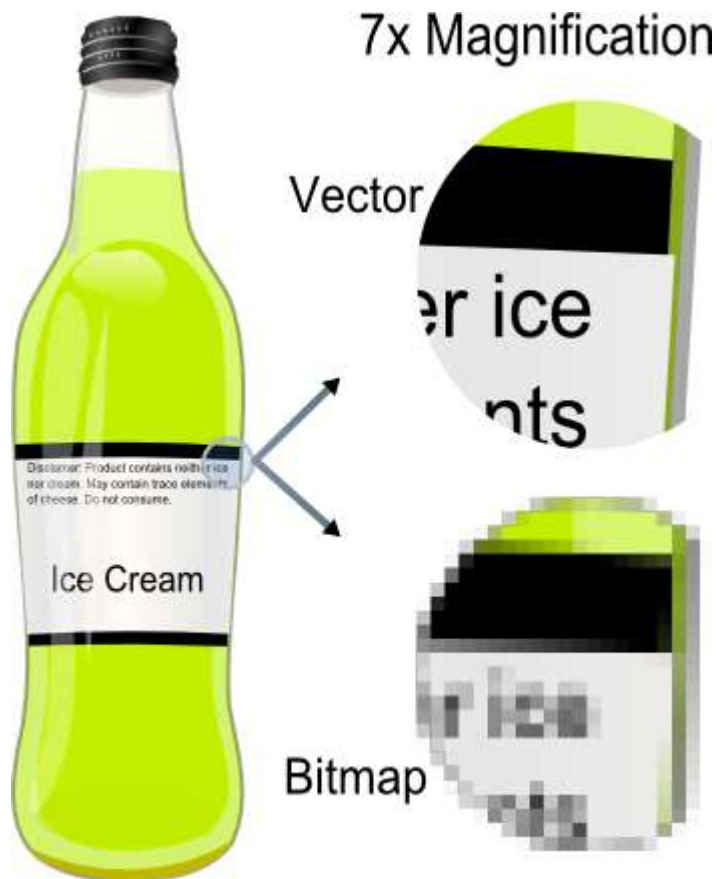


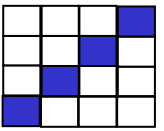
Images



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

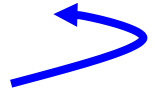
- Vector graphics are stored as a collection of vectors
- Small memory size
- Complex rendering
- Bitmap/Raster Graphics are stored as array of pixels
- Large memory size
- Simple rendering

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

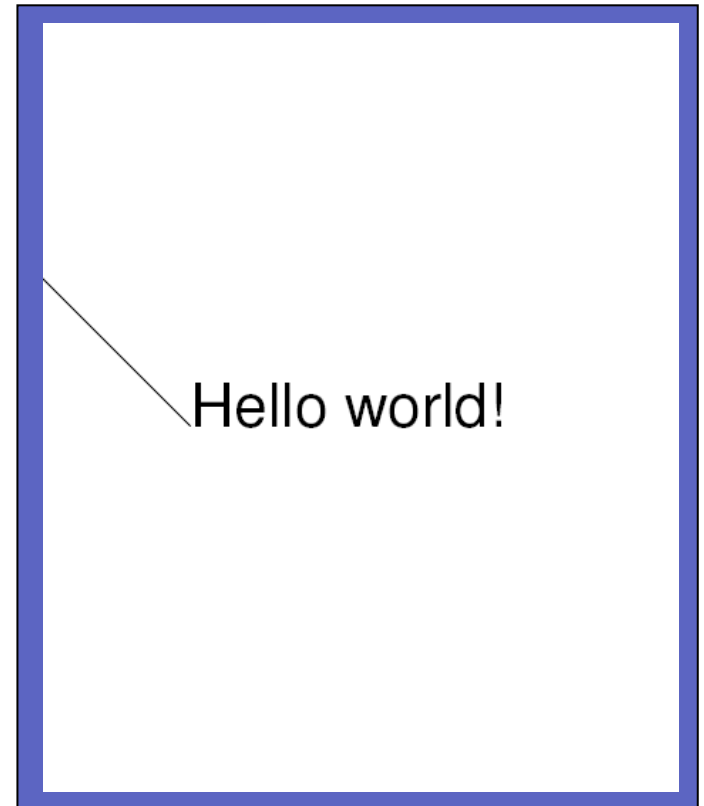
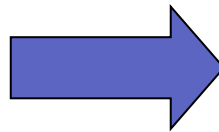


- A vector graphic file format is composed of analytical geometry formula representations for basic geometric shapes (e.g., line, rectangle, ellipse, etc.)
- Used primarily for storing graphics produced with technical drawing programs or "simple images"
- The graphic can be resized easily by simply changing 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.

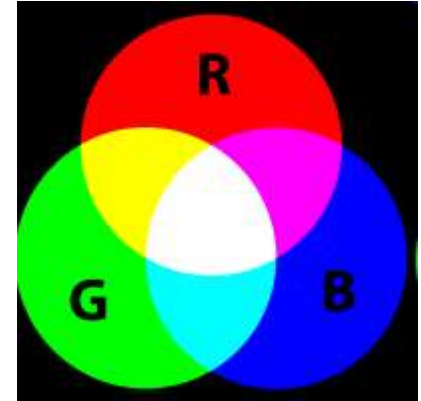


Color Models

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

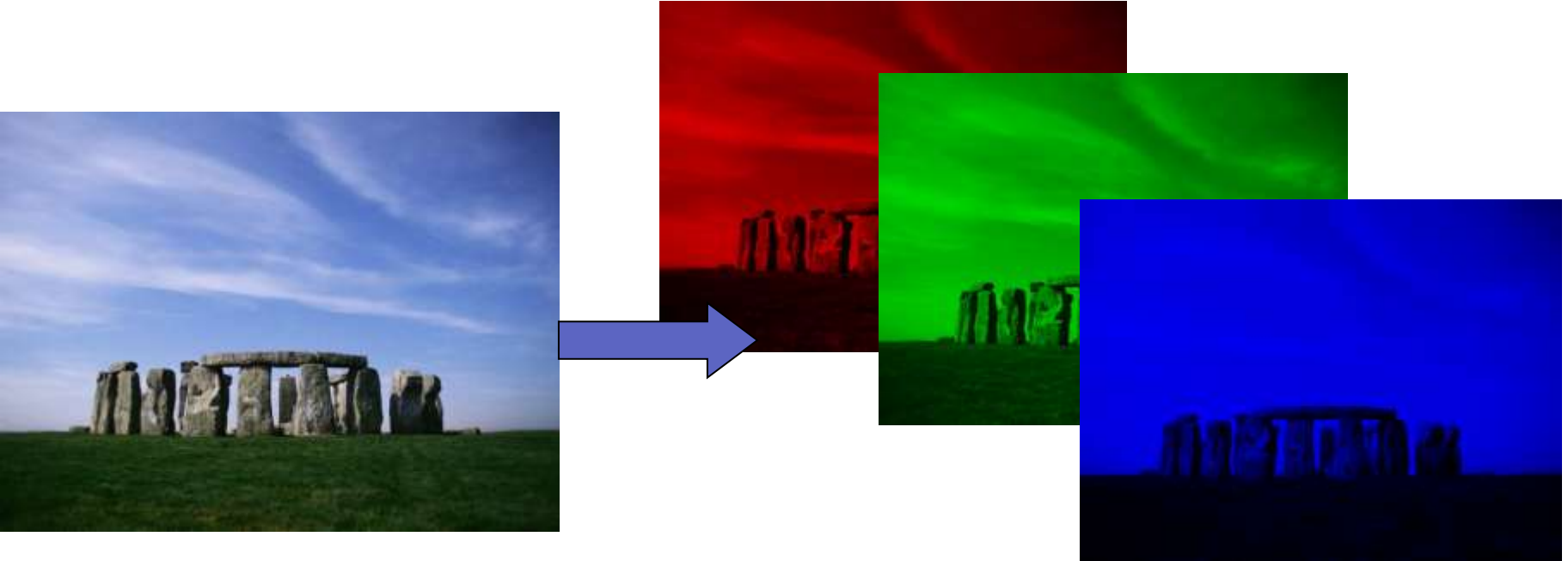
RGB Color Model

- 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
- All colors together 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



Source: en.wikipedia.org

RGB: Splitting-Up Color Channels

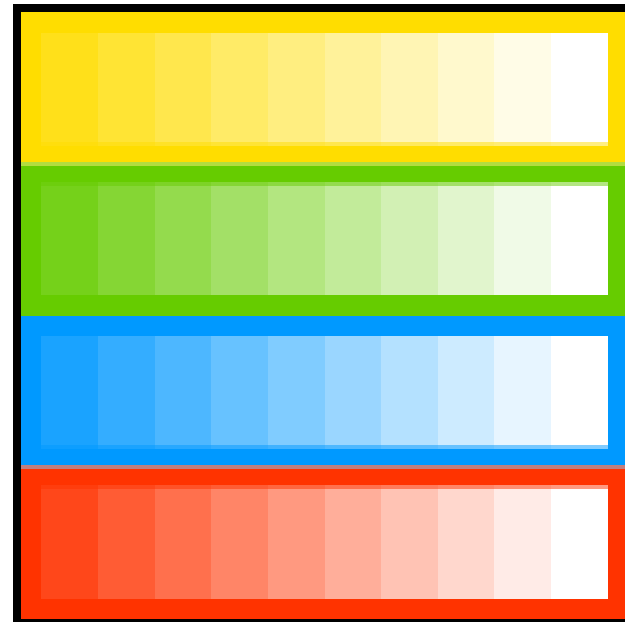


Example: $R + G =$



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

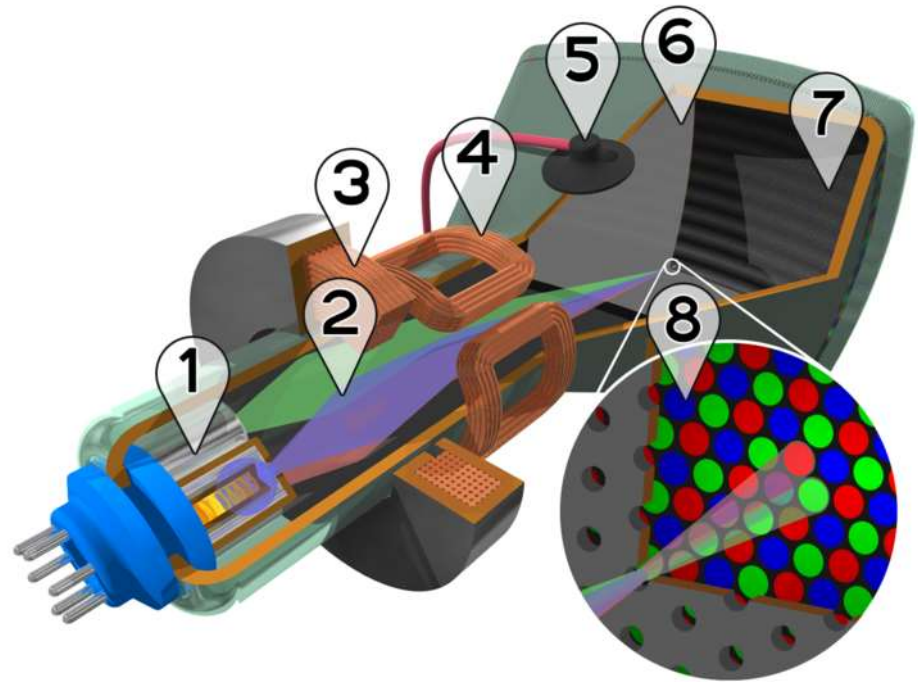
- CMYK is a subtractive color model
- Colors within model: cyan, magenta, yellow, and key (black)
- Colors specified by levels of CMYK
e.g., (255,255,255,255) -> BLACK,
(0,0,0,0) -> WHITE
- Conversion from RGB:
 - $R = 1.0 - (C + K)$
 - $G = 1.0 - (M + K)$
 - $B = 1.0 - (Y + K)$
- CMYK is used for printing (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 different compression algorithms)
- compression ration of 1:15 for "visually lossless" images

■ PNG (Portable Network Graphics)

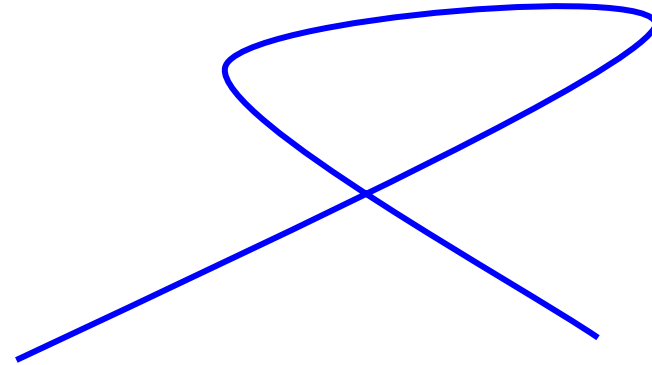
- supports 24-bit colors (16,7 million colors)
- supports transparency
- lossless compression scheme



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
- ...

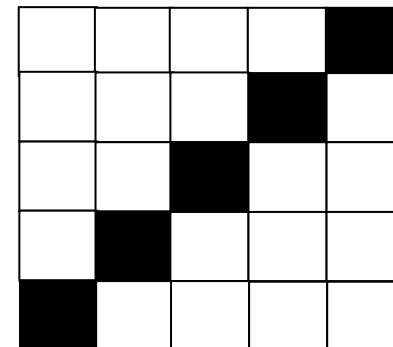
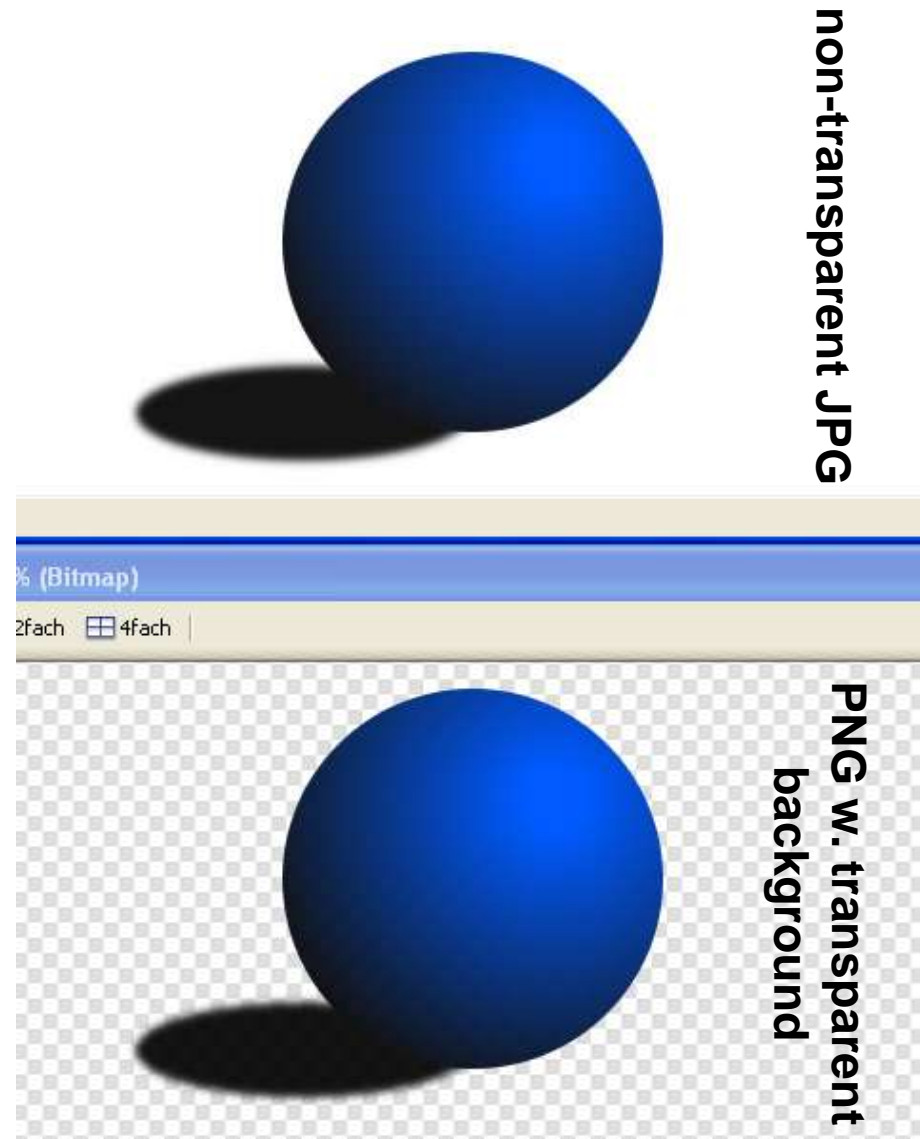


Image Transparency

- 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

```

```

- Specifying image dimensions:

```

```



not recommended
-> CSS3

Image Maps

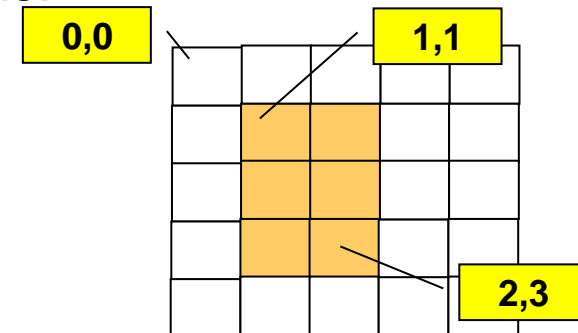
- Allows to specify "hotspots" for a single image file
- Coordinates are defined in pixels (x,y) from the top left corner
- First specify the image to be used as an imagemap

```

```

- 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>
```

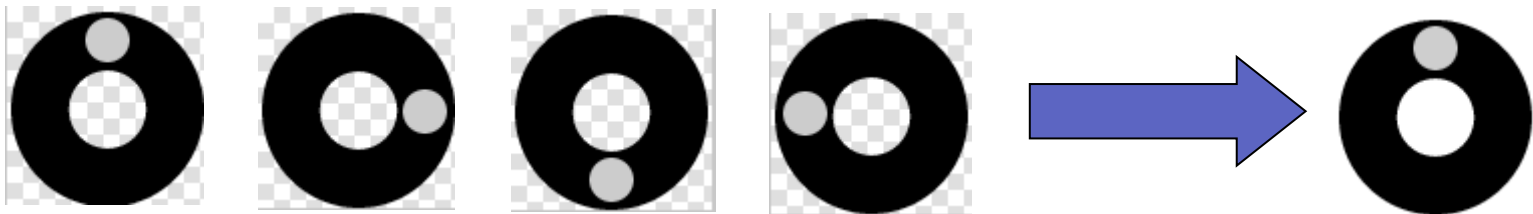


- The following shapes are supported:
 - rect: rectangular area; specified by coordinates of upper-left and lower-right corner
 - circle: circular area; specified by coordinates of center and radius
 - poly: polygon area; coordinates of each point specified

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

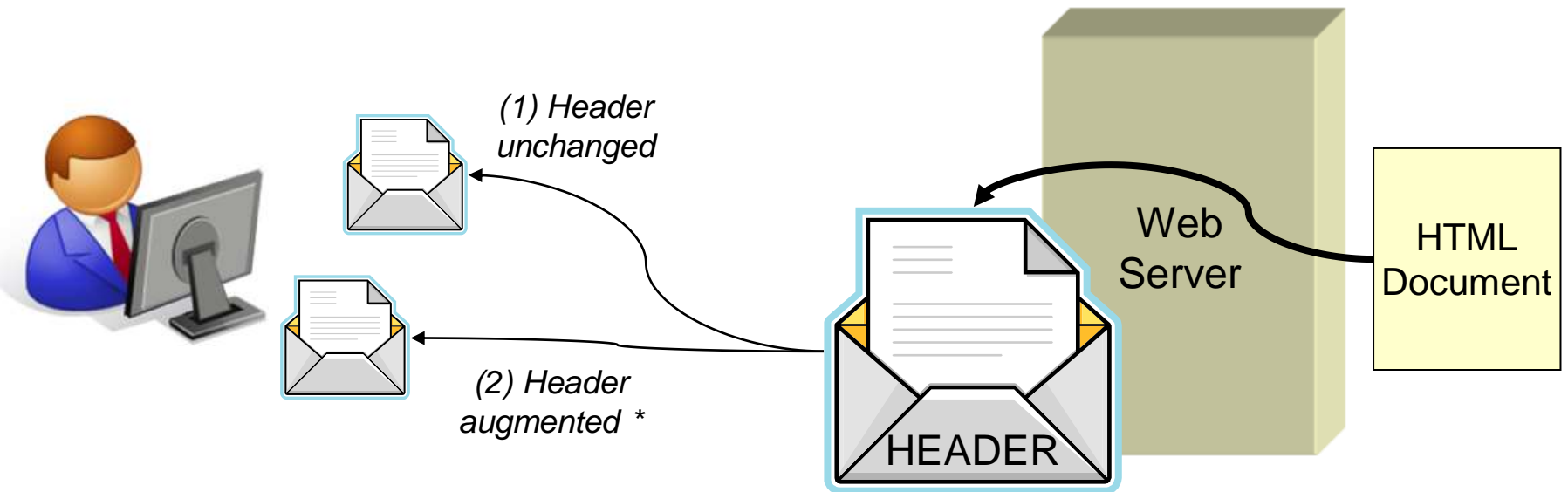


Exercises 2.6 - 2.9

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" >
```

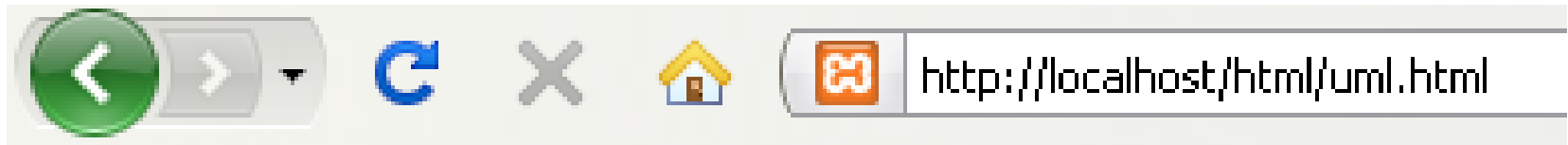
- HTML5

```
<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>
```

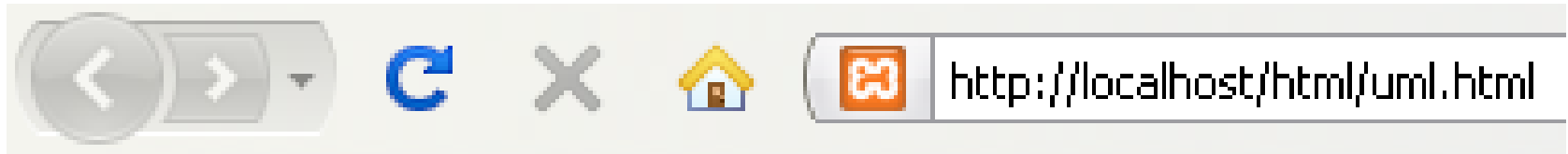


A text with German special characters Ä Ö Ã ¶ Å ¼ Å ÿ Å „ Å – Å œ

A text with German special characters ä ö ü ß Ä Ö Ü

Encoding Examples (XHTML): utf-8

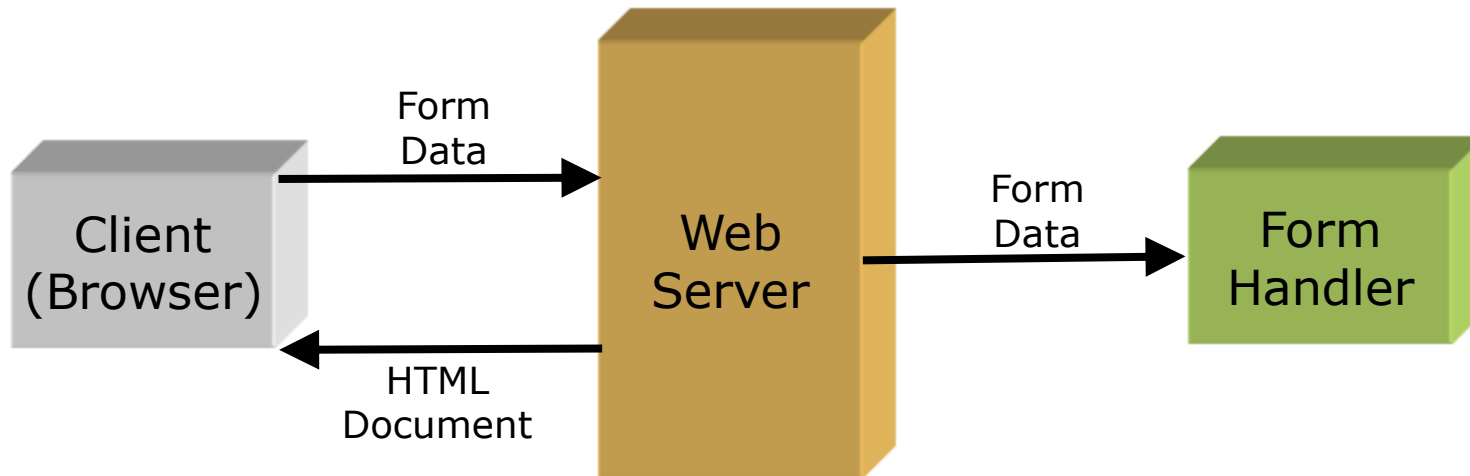
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" >
</head>
<body>
<p> A text with German special characters äöüßÄÖÜ </p>
<p> A text with German special characters &auml;&ouml;&uuml;
&szlig;&Auml;&Ouml;&Uuml; </p>
</body>
</html>
```



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 submits 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:

Name:

■ Password Input Boxes

Password:

Password:

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>
```

Gender: ☐ male ☐ female

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>
```

☐ Onions ☐ cheese

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:

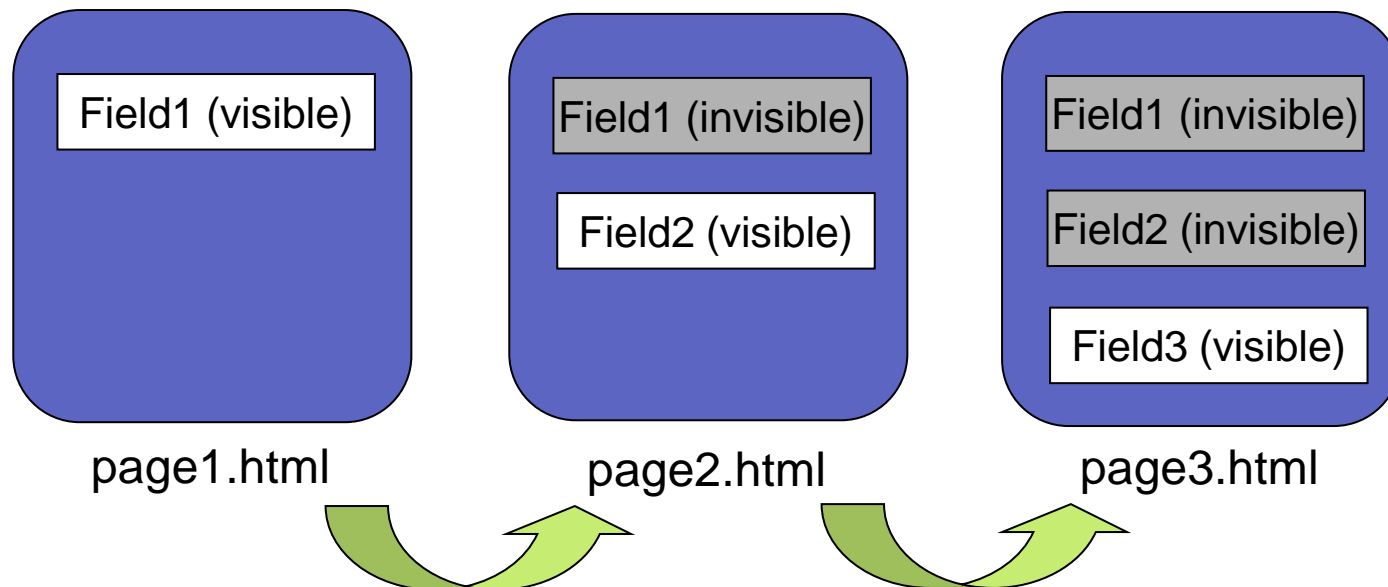
```
<textarea name="text1" cols="50" rows="10">Lorem ipsum dolor  
  sit amet, consectetur adipiscing elit. Maecenas luctus  
  lectus ut pede. Donec aliquet mauris non mi. Nam placerat  
  dolor ultricies arcu. Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit. Nunc faucibus tincidunt  
 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

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"/>`

button_text_submit

button_text_reset

Form Elements

Grouping and Accessibility

- Grouping of form elements via a fieldset

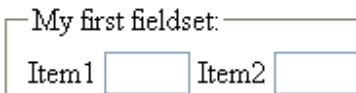
```
<fieldset>
```

```
  <legend>Caption for grouped fields </legend>
```

```
  <input ... />
```

```
  <input ... />
```

```
</fieldset>
```



My first fieldset:

Item1 Item2

- 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

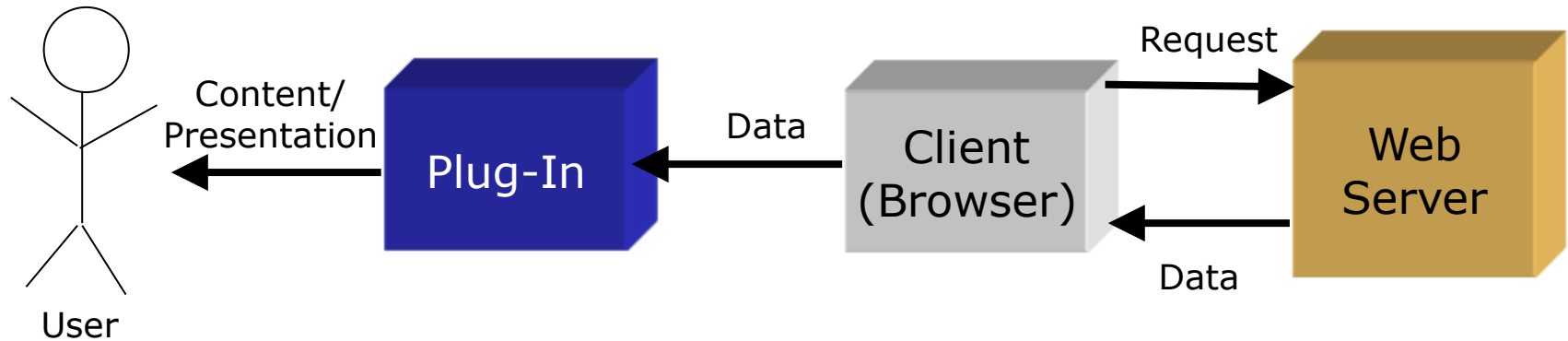
```
<input type="text" name="t1" id="id_t2" value="do_not_modify"
      readonly="readonly" />
```

- 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 in HTML4, "new" in HTML5)**
 - 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/1JwgP44Ap9E">
</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 interpreted the <embed> tag
- **Example (XHTML 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>
```

The <iframe> Tag

Currently Youtube-generated Code:

```
<iframe width="854" height="480"  
src="https://www.youtube.com/embed/Vebx1Ja9zm8" frameborder="0"  
gesture="media" allowfullscreen></iframe>
```

Exercises 3.1, 3.3-3.5



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 Seal](#) by Eric Stoltz

[Make 'em Proud](#) by Michael McAgnon and Scotty Reifsnnyder

[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, consectetur adipiscing elit. </p>
```

```
<p><b><u>Heading Two</u></b></p>
```

```
<p> Lorem ipsum dolor sit amet, consectetur 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, consectetur adipiscing elit. </p>
```

```
<p class="heading">Heading Two</p>
```

```
<p> Lorem ipsum dolor sit amet, consectetur 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>

"Cascading" Styles Sheets

- Styles can be defined by different authorities
 - Author - author of a document specifies style
 - User - the user (viewer) of a document specifies a style
 - User Agent - user agent (Browser) specifies default style sheets (when no other exist)
- "Cascading" stylesheets means that styles can be overwritten/re-defined
- Foreground styles are "inherited" (e.g., styles for the <body> element are inherited by all descendants)
- The following rules of precedence are applied:
 - Author styles override user styles, which override user agent styles
 - More specific declarations override less specific declarations
 - Styles specified last override previously defined styles

Style Selectors

- Selectors are used to match elements in a document
- Selector expression can range from simple elements such as `<h1>` up to complex selectors specifying a sub-element having relationship to other styles
- Properties specify which properties of an element are to be affected by a style (e.g., font, color, ...)
- Property values can be specified by constants (bold, blue), real-word measures (cm, pt), screen measures (px), percentages, color codes (#rrggbb or rgb(r,g,b)), angles, time values (s, ms), ...
- Syntax:

```
selector_expression {  
    property: value(s) ;  
    property: value(s) ;  
    /* ... */  
}
```

Example:

```
h1 {  
    color: blue;  
}
```


CSS: Matching Elements

- *by (tag)name*

- h1, h2, h4 { color: green;}

- *wildcard (universal selector):*

- Matches all elements within a document

- * {color: blue;}

- tr td ol { color: red;}

- Matches all ol elements following a td element following a tr element

- tr * ol {color: blue;}

- Matches all elements between a tr and ol

CSS: Matching Elements

- *by class*

- "Classes" are identifiers within elements
- Definition of "classes" via `<... class="class_id" />`
- Example:

```
<p class="dark_background"> ... </p>
```

- Class as sub-element of `<p>`
 - `p.dark_background { ... }`
- wildcard via `"."` or `"*."` selector
- `.dark_background { ... }` Or `*.dark_background { ... }`

CSS: Matching Elements

- *by Pseudoclass:*

- pseudoclasses do not require an explicit class definition

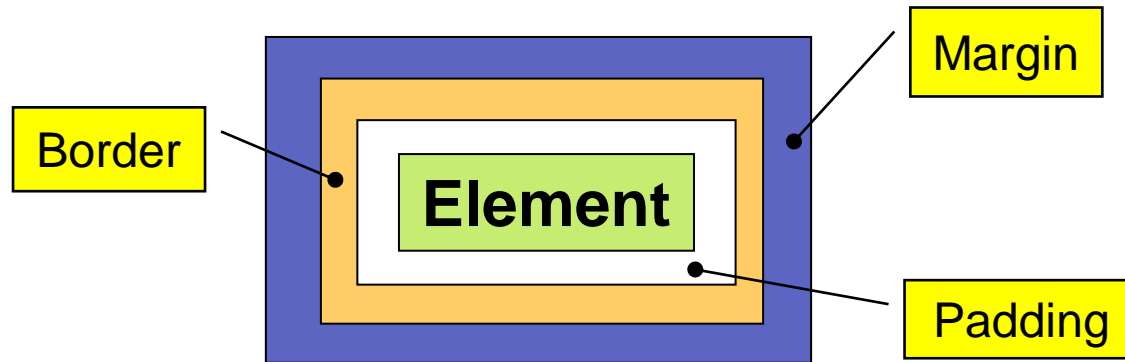
- Anchor Styles:

- :link - unvisited links
- :visited - visited links
- :active - active links
- :hover - mouse over link
- :focus - tab focus on link

- *by IDs:*

- matches all elements with a particular ID
- **#source { ... }**
- **<p id="source"> ... </p>**

CSS Box Model



- by default, all XHTML elements are contained within a box
- the CSS box model has the properties margin, padding, and border
- CSS Box Properties:
 - padding, padding-top, padding-right, padding-left, padding-bottom
 - border-width, border-top-width, border-right-width, border-left-width, border-bottom-width
 - border-top-style, ...
 - None, Hidden, Dotted, Dashed, Solid, Double, Groove, Ridge, Inset, Outset
 - margin, margin-top, ...

CSS Element Positioning

- CSS supports four positioning methods
 - position: static (default mode)
 - position: relative
 - position: absolute
 - position: fixed
 - position: sticky
- Positioning can be defined using the *position* property
- Positioning of elements can replace table-based layouts and Frames

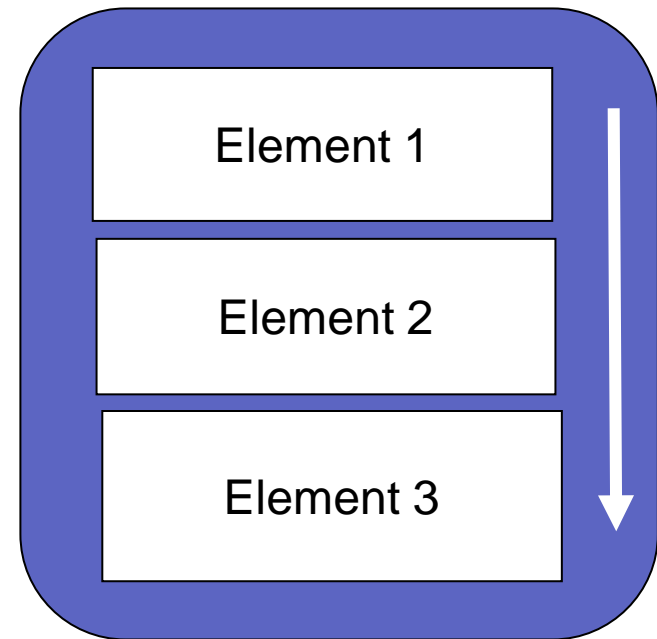
CSS Element Positioning

Static Positioning

- causes elements to be rendered in-line with other elements (default)

- Example:

```
p.pos_static {  
    color: blue;  
    position: static;  
}
```



mypage.html

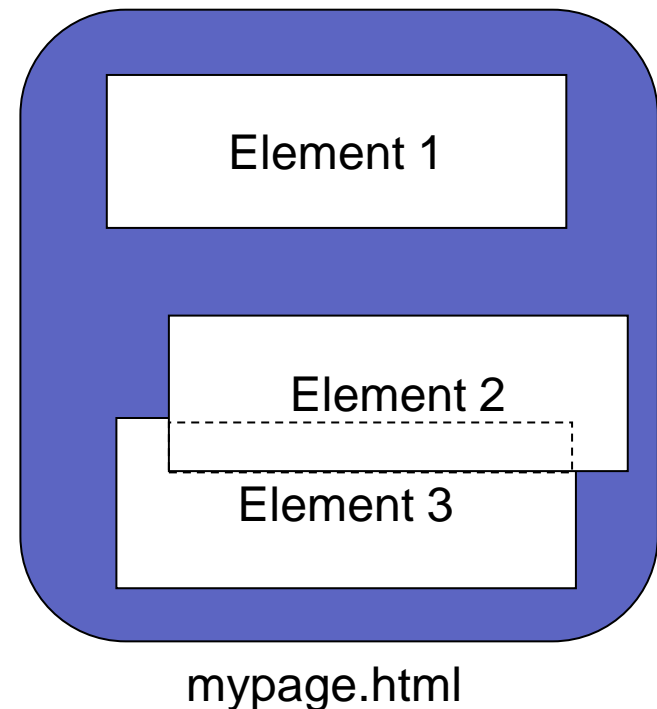
CSS Element Positioning

Relative Positioning

- Positions elements relative to their normal position (not to that of previous elements!)
- Relatively positioned elements may overlap following elements

- Example:

```
p.rel {  
    position: relative;  
    top: 50px;  
    left: 50px;  
}
```



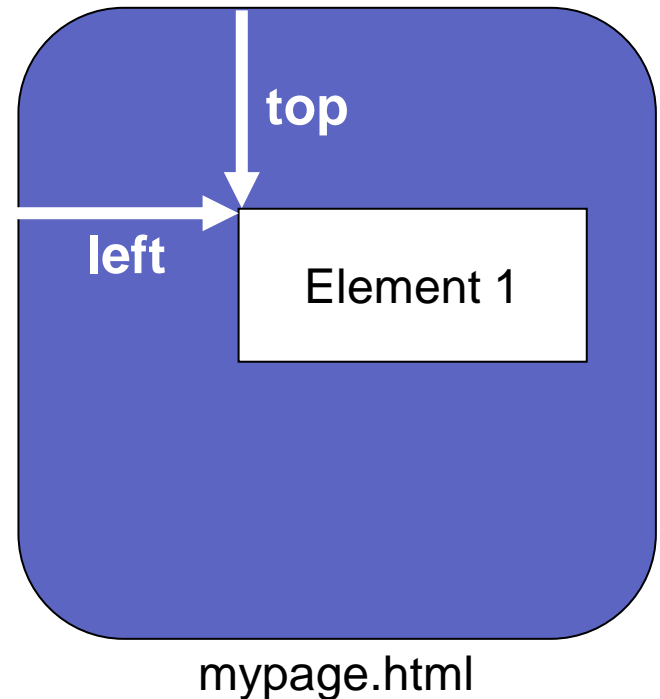
CSS Element Positioning

Absolute Positioning

- Element is positioned absolutely (related to user view)
- Reference point is upper left corner
- Absolute positioning only refers to first positioning; when user scrolls, element is moved accordingly

- Example:

```
p.abs {  
    position: absolute;  
    top: 100px;  
    left: 100px;  
}
```



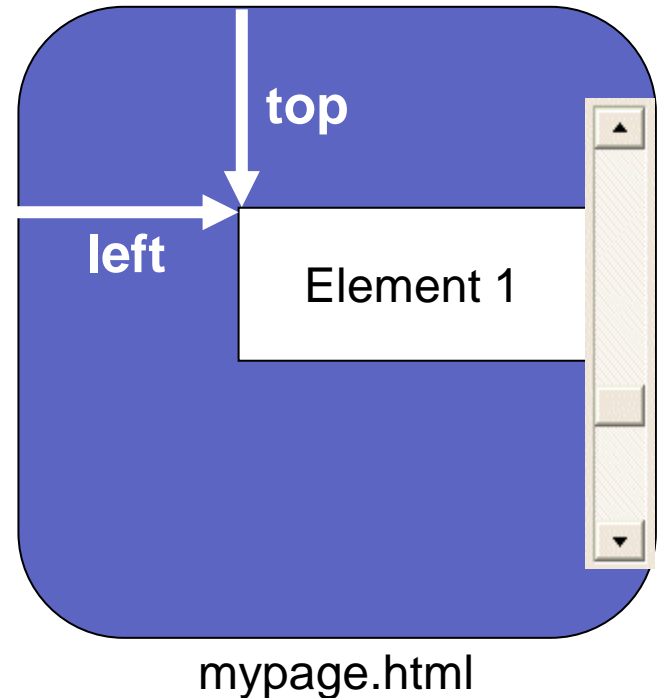
CSS Element Positioning

Fixed Positioning

- Similar to absolute positioning, but elements stays in position even when user scrolls

- Example:

```
p.fix {  
    position: fixed;  
    top: 50px;  
    left: 50px;  
}
```



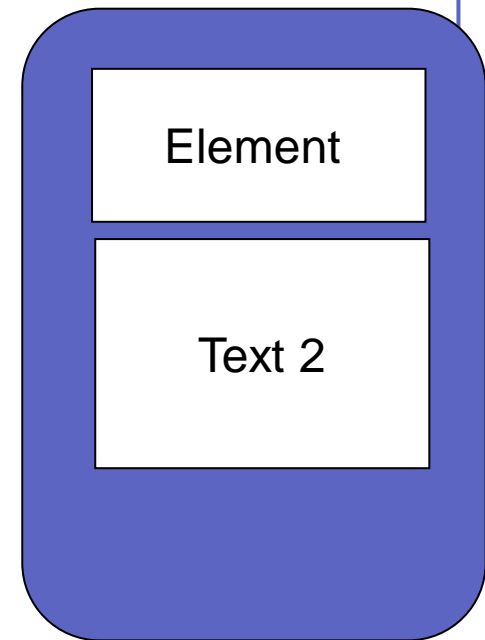
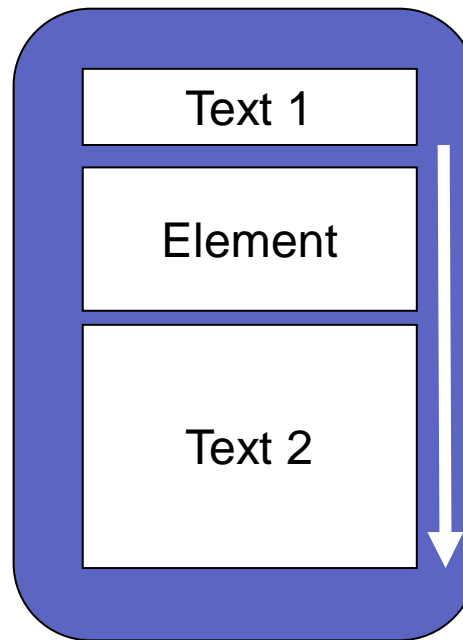
CSS Element Positioning

Sticky Positioning

- positioning relative to the user's scroll position (element toggles between relative and fixed)

- Example:

```
p.pos_sticky {  
  color: blue;  
  position: sticky;  
}
```



CSS Element Layers

- layers introduce 3rd dimension to the GUI
- stacking of elements is defined by z-index property
- z-index: <integer> | auto

```
<html>
<head>
<style type="text/css">
```

```
img.x
{
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: 1;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>This is a Heading</h1>
```

```

```

```
<p>Default z-index is 0. Z-index 1 has higher priority.</p>
```

```
</body>
```

```
</html>
```



This is a Heading

Default z-index is 0. Z-index 1 has higher priority.

Exercises 3.6, 3.7, 3.8