Tutorial 3 - Working with Cascading Style Sheet (CSS)

Cascading Style Sheets (CSS)

- a **style** defines the appearance of a document element.
- a **style sheet:** collection of styles for a Web page or Website
- style sheets use common language and syntax
- main style sheet standard: **Cascading Style Sheets (CSS)**

CSS history

- developed by WWW Consortium (www.w3c.org), the same organization that develops standards for HTML
- designed to augment HTML, not replace it
- a whole new way of formatting Web pages
- provides several tools not available with standard HTML
- different versions
  - CSS1 released in 1996
  - CSS2 released in 1998
  - CSS3, latest standard being developed

Browser support for CSS

- Internet Explorer
  - Internet Explorer 4.0 provides support CSS1
  - Internet Explorer 5.0 provides the best support for CSS1
- Netscape
  - Netscape's support for CSS1 has been spotty
  - Netscape has been pushing their own style sheet language over CSS, however, Netscape 5.0 may end up supporting CSS1

Benefits of style sheets

- a design tool
- makes website more flexible
- easier to maintain and modify
- more aesthetically interesting
- consistent look

Style types

- three ways of employing CSS in Web pages:
  - **inline styles**
    - styles are added to each tag within the HTML file
    - style only affects that particular tag
  - **embedded** or **global styles**
    - applied to an entire HTML file
    - allowing the Web designer to modify the appearance of any tag in the document
  - **linked** or **external style sheets**
    - placed in an external file, linked with Web pages
    - allowing Web designer to modify the appearance of tags in several documents

Using inline styles

- format a single section, better use inline style
- syntax

```
<tag style="style declarations">
```

  - ***tag*** is the name of an HTML element (h1, h2, p, etc)
  - ***style declarations***
    - the styles defined for the particular tag
    - must be enclosed within double quotation marks
    - use semi-colon separate multiple attributes

```
<tag style="attribute1:value1; attribute:value2">
```

  - e.g.

```
<h1 style="color:gold; font-family:sans-serif">
```


Creating an embedded style

- **embedded style**, a style applied to various sections within a Web page
- use `<style>` tag within the head section
- within `<style>` tag, enclose style declarations
- syntax

```
<style type="style sheet language">
  style declarations
</style>
```

  - ***style sheet language*** identifies the type of style language used in the document
  - default is "text/css" for using CSS

Style declarations

- syntax for style declaration:

```
selector{attribute1:value1; attribute2:value2; ...}
```

  - collection of attributes and values also referred to as ***declaration*** of the selector
- **selector**
  - identifies an element in your document, e.g., a heading
  - identifies attributes and values within the braces for that element
- example

```
<style type="text/css">
  h1 {color: gold; font-family: sans-serif}
</style>
```


Defining a embedded style

Grouping selectors

- apply the same declaration to a group of selectors by including all of the selector names separated by commas
- example:

```
<style type="text/css">
  h1, h2, h3, h4, h5, h6 {color:gold; font-
  family:sans-serif}
</style>
```


Using an external style sheet

- creating a text file containing style declarations
- have the extension "**.css**", though this is not a requirement
- within a style sheet, **<style>** tag is NOT needed, only need style declarations

Linking to style sheets with <link>

- general syntax for using the <link> tag

```
<link href="URL" rel="relation_type"
  type="link_type" />
```

  - ***URL*** is the URL of the linked document
  - ***relation_type*** establishes relationship between linked document and Web page
    - rel="stylesheet" for linking to a style sheet
  - ***link_type*** indicates language used in linked document
    - type="text/css" for linking to a style sheet
- example

  link to a style sheet named "**farm.css**,"

```
<link href="farm.css" rel="stylesheet" type="text/
  css" />
```


Linking to style sheets with @import

- enclose the **@import** command within the embedded **<style>** tags

```
<style type="text/css">
   @import url(company.css)
   @import url(company.css)
</style>
```

  - ***company.css*** is the URL of the style sheet file
- limited browsers support @import
- better off using the **<link>** tag

Style precedence

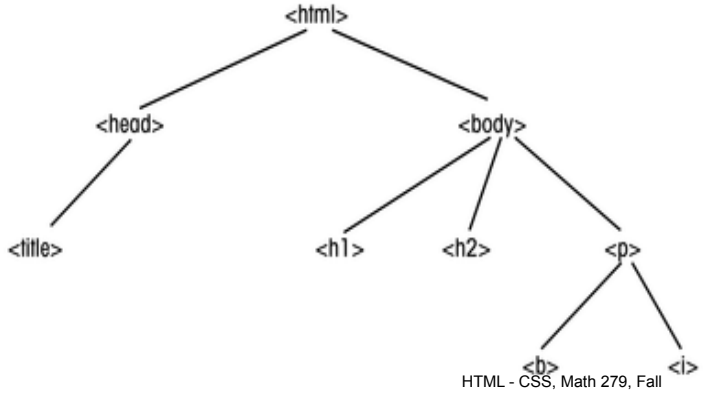
- in cases styles conflict, precedence is determined in the following order:
  - an **inline style** overrides any **embedded style** or **external style sheet**
  - an **embedded style** overrides an **external style sheet**
  - an **external style** sheet overrides the **internal style rules** set by the Web browser

Changing styles

- as a change is made to a style at one level, the changes are cascaded through to the other levels ➔ **cascading style sheets**
- need to keep track of the inline, embedded, and external style sheets to correctly predict the impact that style changes have on the appearance of each page

Working with style inheritance

- Web pages invariably have elements (e.g., tags) placed within other elements
- sample tree structure of Web elements



Parent and descendant elements

- an element that lies within another element is called a **descendant** or **descendant element**
  - e.g., in previous slide, <b> tag is a descendant of <p> tag
- an element that contains another element is called the **parent** or **parent element**.
  - e.g., **<body>** tag is the parent of all other tags used to format the Web contents
- using the principle of **inheritance**, *styles defined for a parent tag are transferred to its descendants tags*

Contextual selectors

- use tree structure to better control styles
  - apply a style only to *direct descendant* of a parent element, use syntax: `e1 e2`
  - ***e1*** and ***e2*** are the names of HTML elements (i.e. tags) and e2 is directly below e1 in the tree structure of elements
  - example: `li b {color:blue}`

    only changes the color of the *boldface* text residing within a <li> tag to *blue*
- not all browsers support contextual selectors

Generally used attributes and their values

Using font families

- The **font-family** attribute allows you to choose a font face for Web content
- CSS works with two types of font faces:
  - **specific font**, which is a font such as **Arial**, **Garamond**, or **Times New Roman** that is actually installed on a user's computer
  - **generic font**, which is a general description of a font, allowing the operating system to determine which installed font best matches it
  - CSS supports five **generic** font types: **serif, sans-serif, monospace, cursive, and fantasy**

Generic fonts

Generic Names	Font Samples		
serif	defg	defg	defg
sans-serif	defg	defg	defg
monospace	defg	defg	defg
cursive	defg	defg	defg
fantasy	DEFG	DEFG	defg

within each generic font there can be a wide range of appearances

Generic, specific fonts issues

- using generic fonts cannot be sure which specific font used
- specific fonts are preferred
- provide several fonts to choose from
- list specific font names first, followed by a generic font name, in case none of specific fonts can be found

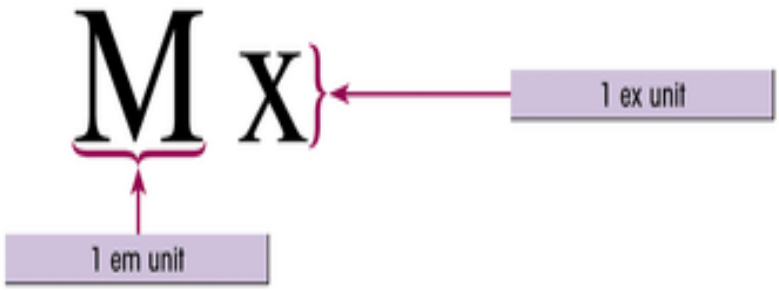
Managing font size

- in CSS, use `font-size` attribute to manage font sizes
- font sizes can be expressed
  - as a unit of length
  - with a keyword description
  - with a keyword expressing the size relative to the font size of parent element
  - as a percentage of the parent element

Absolute and relative units

- use unit of length, **absolute units** or **relative units**
  - **absolute units** define the font size based on standard units of measurement:
    - mm (millimeter),
    - cm (centimeter),
    - in (inch),
    - pt (point)
    - pc (pica)
  - **relative units** express the font size relative to a size of a standard character
    - **em unit** is equal to the *width* of capital letter "**M**" in browser's default font size
    - **ex unit** is equal to the *height* of a small "**x**" in default font size

em, ex units



Pixels

- a pixel is the smallest element recognized by the monitor
- text that is 10 pixels high may be perfectly readable at a low-resolution (e.g., 640 x 480) monitor, but it can become unreadable at high-resolution (e.g., 1024 x 768) monitor

Descriptive keywords

- seven descriptive keywords for font size
  - xx-small
  - x-small
  - small
  - medium
  - large
  - x-large
  - xx-large
- Example

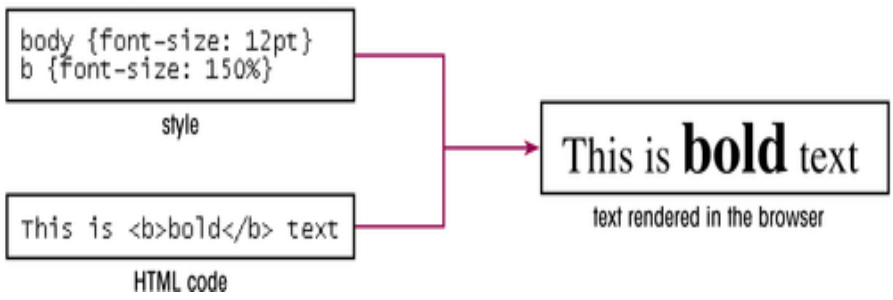
```
body {font-size: medium}
```

Keywords: smaller, larger

- using keywords "**larger**" and "**smaller**,"
  - makes the font **one size** larger or smaller than the size of parent element
  - example: to make h2 heading one size larger than the body text, you could use the following style

```
body {font-size: medium}
h2 {font-size: larger}
```


Font size as percentage of parent tag



The font size of bold content (defined by <b> tag) is 150% of the size of surronding text (where the font size is defined by <body> tag)

Specifying letter, word spacing

- set the space between individual letters, **letter-spacing** attribute

  `letter-spacing: size`

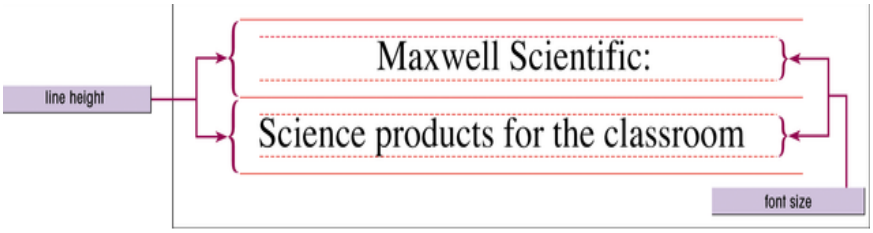
- change the spacing between individual words, **word-spacing**

  `word-spacing: size`

- ***size*** can be
  - the value "**normal**", which allows the browser to determine letter spacing
  - a number expressed in *inches, millimeters, centimeters, em units*, etc.
    - the same measuring units used to describe font size

Specifying line height

- **line-height** attribute modifies vertical space between lines of text, specifies the minimum distance between the baselines of adjacent lines.
- example



Specifying line height (cont.)

- syntax

  `line-height: size`

  - ***size*** is either a specific length, a percentage of the font size, or a number representing the ratio of the line height to the font size
  - standard ratio is 1.2, which means that the line height is 1.2 times the font size
  - example, to make paragraphs double-spaced use the style definition

    ```
    p {line-height: 2}
    ```


Font weights

- CSS considers "**bold**" to be an aspect of the font's weight, or line thickness.
- **font weights** can be expressed as *an absolute number ranging in intervals of 100, going from 100 (the lightest) up to 900* (the heaviest or "most bold").
- for most fonts,
  - a weight of 400 corresponds to normal text
  - a weight of 700 can be used for bold text
  - a weight of 900 for "extra" bold text
- example

  ```
  h2 {font-weight: 700}
  ```

Aligning text horizontally and vertically

- **text-align** attribute

```
text-align: alignment
```

  - ***alignment*** can be left, center, right, or justify
  - setting the text-align value to "**justify**" stretches the text, extending it from the left to the right margin
- **vertical-align** attribute

```
vertical-align: alignment
```

  - ***alignment*** has one of the keyword values

Values for vertical-align

ATTRIBUTE VALUE	DESCRIPTION
baseline	Aligns the element with the baseline
bottom	Aligns the bottom of the element with the bottom of the lowest element (text or image) in the line
middle	Aligns the element in the middle of the text
sub	Aligns the element as a subscript
super	Aligns the element as a superscript
text-bottom	Aligns the element with the font's bottom
text-top	Aligns the element with the top of the tallest letter
top	Aligns the element with the top of the tallest element (text or image) in the line

Special text attributes

- CSS provides three attributes for special text effects:
  - text-decoration
  - text-transform
  - font-variant

text-decoration attribute

- attribute name: **text-decoration**
- values
  - none
  - underline
  - overline
  - line-through
- examples

Maxwell Scientific teaches science
text-decoration:none

Maxwell Scientific teaches science
text-decoration:underline

Maxwell Scientific teaches science
text-decoration:overline

~~Maxwell Scientific teaches science~~
text-decoration:line-through

text-transform attribute

- attribute name: **text-transform**
- attribute value:
  - **capitalize**
    capitalize the first letter of each word in a paragraph
  - **uppercase**
    display the text in all capital letters
  - **lowercase**
    display the text in all lowercase letters

text-transform attribute, example

Maxwell Scientific teaches science
text-transform:none

Maxwell Scientific Teaches Science
text-transform:capitalize

MAXWELL SCIENTIFIC TEACHES SCIENCE
text-transform:uppercase

maxwell scientific teaches science
text-transform:lowercase

font-variant attribute

- **font-variant** attribute create small caps
- small caps are capital letters that are the same size as lowercase letters
- syntax

```
font-variant: small-caps
```

- **Netscape** does not support the font-variant attribute in versions prior to 6.0

font-variant attribute, example

Maxwell Scientific teaches science

font-variant:normal

MAXWELL SCIENTIFIC teaches science

font-variant:small-caps

background color

- change the background color for any element, use the **background-color**
- example

```
blockquote {background-color: silver}
```


RGB colors

- Numerical expression for colors, Red Green Blue – RGB triplet
  - (*<0-255>*, *<0-255>*, *<0-255>*)
  - 0, absence of color
  - 255, highest color intensity
  - e.g., (255,255,255) – white
    (0,0,0) – black
  - 256^3 colors can be represented using RGB triplet

Hexadecimal RGB expression

- Hexadecimals, base-16 numbers

decimal, base 10	0 1 2 3 4 5 6 7 8 9	10	11	12	13	14	15
hexadecimal, base 16	0 1 2 3 4 5 6 7 8 9	A	B	C	D	E	F

- multiples of 16 + remainder
  - e.g., $255_{\text{(in decimal)}} = 15*16+15_{\text{(in decimal)}} = FF_{\text{(in hexadecimal)}}$
- hexadecimal provides short expression for RGB color
  - (255,255,255) = (FF, FF, FF)

Using hexadecimals, colors are represented by 6 digits following a # symbol, e.g., #33CC33

Using number identify *color*

- *e.g.,*

```
h2 { color: rgb(0,154,0);
     background-color: #FF00AA }
```

- more colors can be identified by numbers than by color names
- useful links for RGB color chart
  - find them at class webpage

Enhancements to Color in CSS3

- CSS3 also supports the Hue Saturation Lightness (HSL) model that describes colors based on **hue**, **saturation**, and **lightness**

```
hsl(hue, saturation, lightness)
```

- CSS3 also allows page designers to augment RGB and HSL color values by specifying a color's opacity. Opacity defines how much of the colors below the surface of the current object show through to affect its appearance

```
rgba(red, green, blue, opacity)
hsla(hue, saturation, lightness, opacity)
```


List styles

- **`list-style-type`** attribute chooses the type of label to display alongside text formatted with the **`<ul>`**, **`<ol>`**, or **`<li>`** tags.
- Values for **`list-style-type`** attribute
  - disc, circle, square
  - decimal, decimal-leading-zero, lower-roman, upper-roman, lower-alpha, upper-alpha

Values of list-style-type

list-style-type	Marker (s)
disc	●
circle	○
square	□
decimal	1, 2, 3, 4, ...
decimal-leading-zero	01, 02, 03, 04, ...
lower-roman	i, ii, iii, iv, ...
upper-roman	I, II, III, IV, ...
lower-alpha	a, b, c, d, ...
upper-alpha	A, B, C, D, ...
none	no marker displayed

Creating nested lists

```
ol          {list-style-type: upper-roman}
ol ol       {list-style-type: upper-alpha}
ol ol ol    {list-style-type: decimal     }
ol ol ol ol {list-style-type: lower-alpha}
```

styles

```
<ol>
   <li>Main Point 1</li>

   <ol>
      <li>Subpoint 1</li>
      <li>Subpoint 2</li>

      <ol>
         <li>Item 1</li>
         <li>Item 2</li>

         <ol>
            <li>Subitem 1</li>
            <li>Subitem 2</li>
         </ol>

         <li>Item 3</li>
      </ol>

      <li>Subpoint 3</li>
   </ol>

   <li>Main Point 2</li>
   <li>Main Point 3</li>
</ol>
```

HTML code

I. Main Point 1
   A. Subpoint 1
   B. Subpoint 2
      1. Item 1
      2. Item 2
         a. Subitem 1
         b. Subitem 2
      3. Item 3
   C. Subpoint 3
II. Main Point 2
III. Main Point 3

text as rendered by browser

`list-style-image` attribute

- `list-style-image` attribute creates with a user-defined image file
- syntax

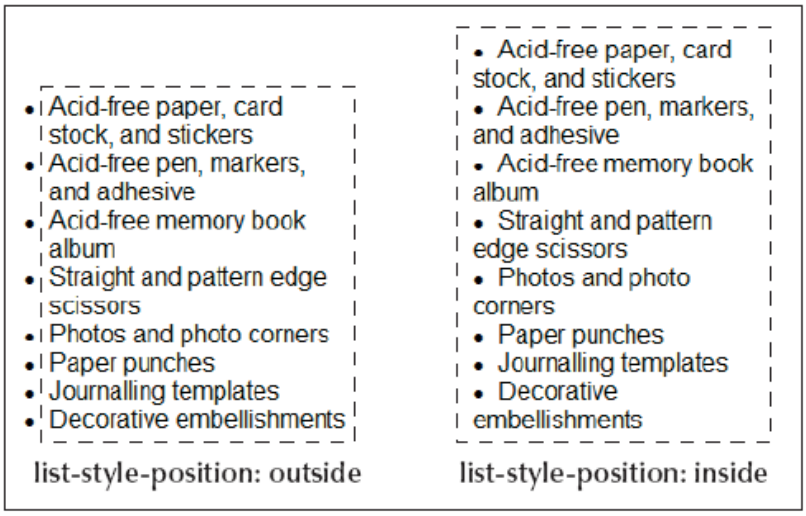
  `list-style-image: url(`***URL***`)`
  - ***URL*** is the location and the filename of the image file
- better include `list-style-type` attribute along with `list-style-image` attribute.

Defining list style position

- list items are treated by CSS as if they have an invisible box around them.
- use `list-style-position` to specify the location of list item labels
- syntax

  `list-style-position:` ***`location_value`***
  - *location_value* is either "**inside**" or "**outside**" (default)

Defining the position of list labels



`list-style` attribute

- `list-style` attribute combines list attributes
- syntax

  `list-style: list-style-type list-style-image list-style-position`

  - *list-style-type*, *list-style-image*, and *list-style-position* are the attribute values for each of the individual list style attributes

The `class` attribute

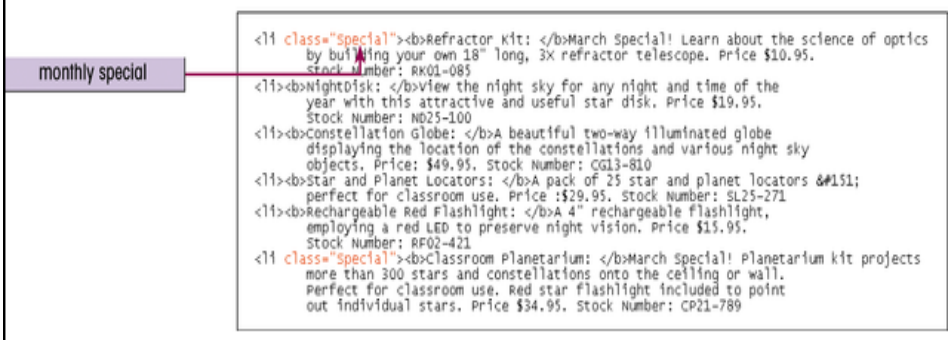
- Web designers may create customized classes by adding the **class** attribute to HTML tags.
- syntax

```
.class_name {styles}
<element class="class_name">
```

  - ***element*** is an HTML tag
  - ***class_name*** is the name of the class
  - meanwhile, define styles for the new created class in the embedded style sheet
- Example

```
.subtitle {color: blue}
h2.subtitle {color:red}
<h2 class="subtitle"> Getting Started</h2>
```


Example



- put `class="Special"` in the first and the last <li> tag of the file astrotxt.html
- meanwhile, put `li.Special {color: red}` in the embedded style sheet

Formatting hypertext links

- A hypertext link can be in one of four states:
  - the link's target has already been visited by the user
  - the link's target has never been visited by the user
  - the link is currently being clicked by the user
  - the user's mouse pointer is hovering over the link
- CSS provides a different selector for each condition.
  - `a:visited {styles for previously visited targets}`
  - `a:link {styles for targets that have never been visited}`
  - `a:active {styles for links that are currently being clicked}`
  - `a:hover {styles when the mouse cursor is hovering over the link} - this is called a rollover effect`
- example

```
a:visited {color:red}
a:hover{color:gold; text-transform:uppercase}
```


Pseudo-Classes and Pseudo-Elements

- A pseudo-class is a classification of an element based on its current status, position, or use in the document

```
selector:pseudo-class {styles;}
```

Dynamic pseudo-classes

Dynamic pseudo-classes

Pseudo-Class	Description	Example
link	The link has not yet been visited by the user.	a:link {color: red;}
visited	The link has been visited by the user.	a:visited {color: green;}
active	The element is in the process of being activiated or clicked by the user.	a:active {color: yellow;}
hover	The mouse pointer is hovering over the element.	a:hover {color: blue;}
focus	The element has received the focus of the keyboard or mouse pointer.	input:focus {background-color: yellow;}

Structural pseudo-classes

Structural pseudo-classes

Pseudo-Class	Matches
root	The top element in the document hierarchy (the html element)
empty	An element with no children
only-child	An element with no siblings
first-child	The first child of the parent element
last-child	The last child of the parent element
first-of-type	The first element of the parent that matches the specified type
last-of-type	The last element of the parent that matches the specified type
nth-of-type(*n*)	The n^{th} element of the parent of the specified type
nth-last-of-type(*n*)	The n^{th} from the last element of the parent of the specified type
only-of-type	An element that has no siblings of the same type
lang(*code*)	The element that has the specified language indicated by *code*
not(*s*)	An element not matching the specified selector, *s*

Pseudo-Elements

Pseudo-elements

Pseudo-Element	Description	Example
first-letter	The first letter of the element text	p:first-letter {font-size:200%}
first-line	The first line of the element text	p:first-line {text-transform: uppercase}
before	Content inserted directly before the element	p:before {content:"Special! "}
after	Content inserted directly after the element	p:after {content:"eof"}

Homework

- Finish reading Tutorial 3 and Review Assignment

background color (recap)

- change the background color for any element, use the **background-color**
- example

```
blockquote {background-color: silver}
```

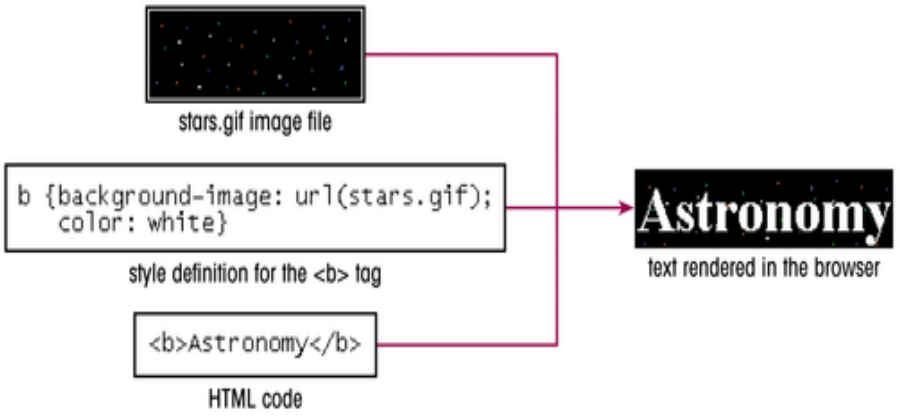

background-image

- background image has four attributes:
  - the source of the image file
  - how the image is repeated in the background
  - where the image is placed on the background
  - whether the image scrolls with the display window
- To specify which file to use for a background, use the syntax:

```
blockquote {background-image: url(URL)}
```

  - ***URL*** is the location of the image file

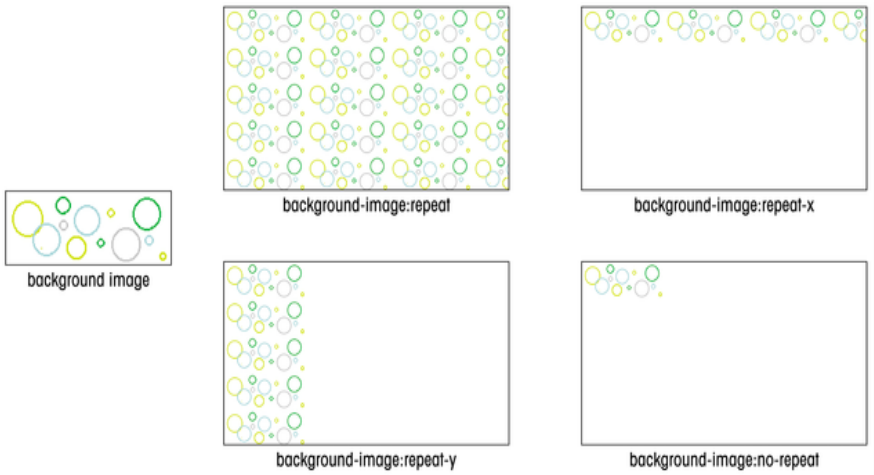
applying a background image to an element



background images (cont.)

- by default, background images are tiled both horizontally and vertically behind the element until the entire element is filled
- control the way the tiling occurs using **background-repeat** style attribute with four values:
  - **repeat,** the image is tiled both horizontally and vertically until the element is filled
  - **repeat-x**, the image is tiled only horizontally across the width of the element
  - **repeat-y**, the image is tiled only vertically across the width of the element
  - **no-repeat**, the image is not repeated at all

background-repeat values, example



background-position attribute

- **background-position** attribute define the position of a background image
  - by default, a background image is placed in the *upper-left* corner of an element
- syntax

  background-position: *value1, value2*
  - *value1* indicates the distance from the left margin
  - *value2* indicates the distance from the top margin
  - these two values are expressed as a percentage of the display area, in units of length, or with keywords.
- example,

```
h1 {background-image: url(m20.jpg);
background-position: 30%, 50%}
```

background-attachment attribute

- **background-attachment** attribute defines whether background images should be moved along with contents
- syntax

```
background-attachment: value
```

  - value can be
    - "**scroll**," to scroll the image along with the element
    - "**fixed**," to place the image in a fixed place in the browser's display window
- default value is "scroll"
- example

```
body {background-color: yellow; background-
image: url(logo.gif); background-attachment:
fixed}
```


background attribute

- **background** attribute combines all of the various attributes for backgrounds
- syntax

```
background: background-color-value background-
  image-value background-repeat-value
  background-attachment-value background-
  position-value
```

- *example*

body {font-color: #330033; background: #FFCCFF url(logo.gif) no-repeat fixed center center}

multiple background images

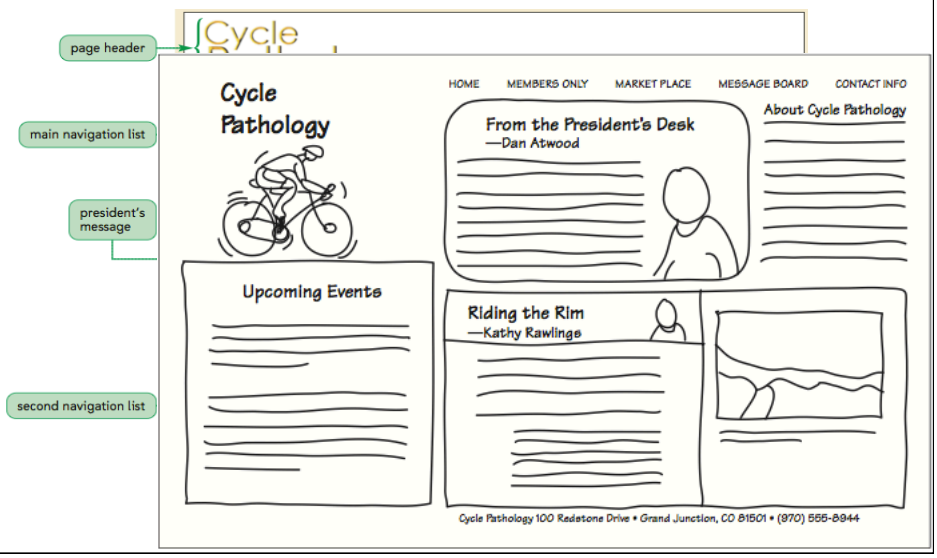
- syntax

  background-property: value1, value2, … ;

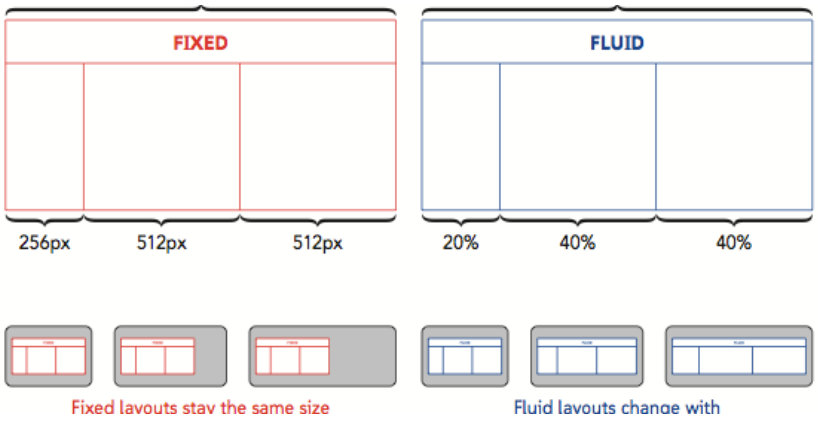
- examples:

```
header { background-color: yellow;
background-image: url(logo.png), url(logo2.png);
background-position: top left, bottom right;
background-repeat: no-repeat; }
header { background:
url(logo.png) top left no-repeat,
url(logo2.png) bottom right no-repeat yellow; }
```


Webpage Layout Design



Fixed layout vs. Fluid layout



Fixed layout vs. Fluid layout

- Fixed layout
  - pros: easy for designer, works well for fixed size object, still readable in worst case
- Fluid layout
  - pros: easy for user, respond well to different devices, consistent look

Setting page size

- Fixed layout
  - Use width and height attributes
  - e.g.: body {width:1024px; height:800px}
- Fluid layout
  - Use percentage for width
  - set upper or lower bound for width and height with min-width, max-width, min-height, max-height
  - e.g.: body {width:95%; min-width:800px; max-width:1000px}

Display style: a revisit

- Two types of elements
  - block vs. inline
- display attribute

  display: type

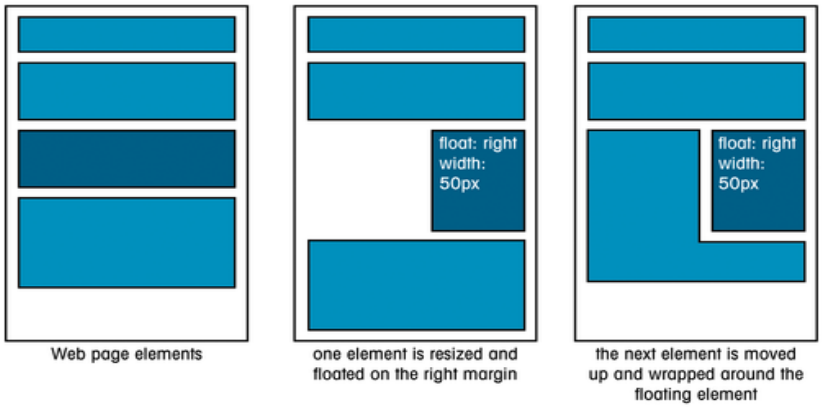
Display Value	Effect On Element
block	Displayed as a block
inline	Displayed in line within a block
inline-block	Treated as a block placed in line within another block
run-in	Displayed as a block unless its next sibling is also a block, in which it is displayed in line, essentially combining the two blocks into one
inherit	Inherits the display property of the parent element
list-item	Displayed as a list item along with a bullet marker
none	Prevented from displaying, removing it from the page structure

The `float` attribute

- the `float` attribute works like the **align**="**left**" or **align**="**right**" attributes used with the `<img>` tags
- this attribute places the block-level element on the left or right margin of the parent element
- syntax:
  - float: *position*

    where *position* can be none, right, or left
  - e.g., `#promoimage {float:right}`

Floating a block-level element

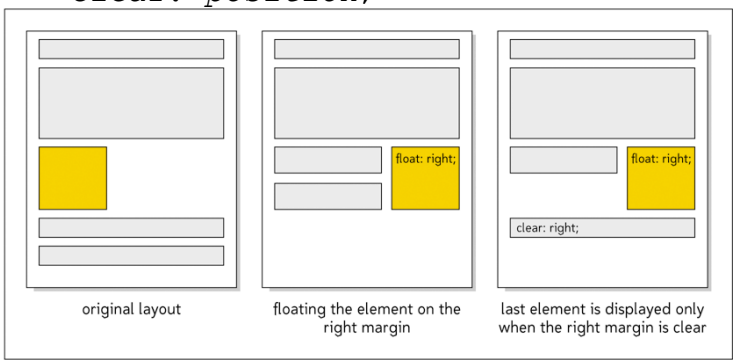


Web page elements | one element is resized and floated on the right margin | the next element is moved up and wrapped around the floating element

Floating Elements

- Clearing a float

```
clear: position;
```

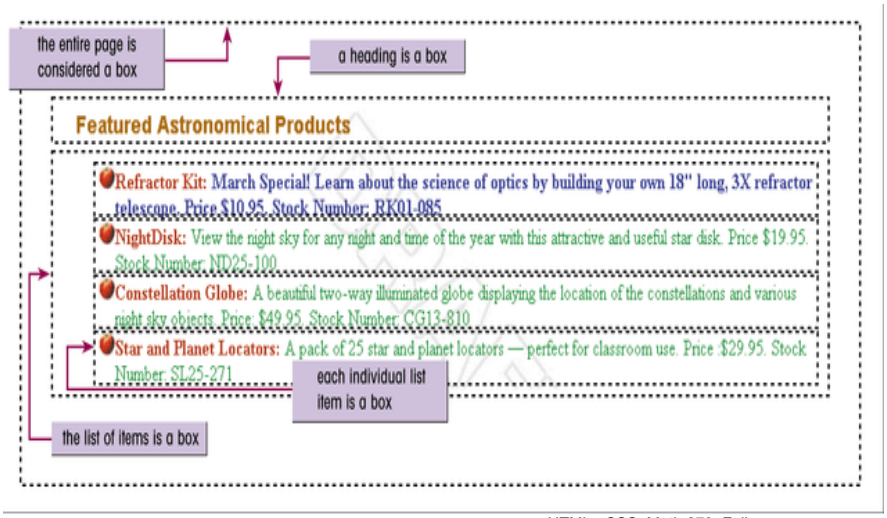


Formatting block-level element – using box model

- With CSS, you can control the layout of a Web page by manipulating the size and location of **block-level elements**.
- **Box Model** describes the structure of block-level elements as they are laid out on the web page
- sample tags that can be treated as block-level elements are:

  <h1> - <h6> , <p>, <blockquote>, <address> , <ul>, <ol>, <dl> , <li>, <dt>, <dd>, , <body>, <hr>, <img>

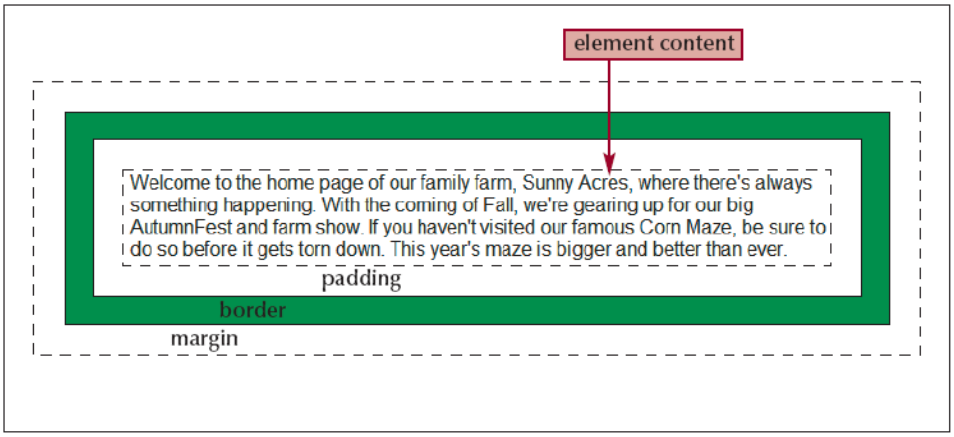
Some block-level elements



Features of block-level element box

- Four features:
  - **margin** between the box and the parent element
  - **border** of the box
  - **padding**, which is the space between the box around the block-level element and the border
  - **content** of the element itself
- CSS provides attributes you can use to control the appearance and behavior of each of these features.

Working with the Box Model



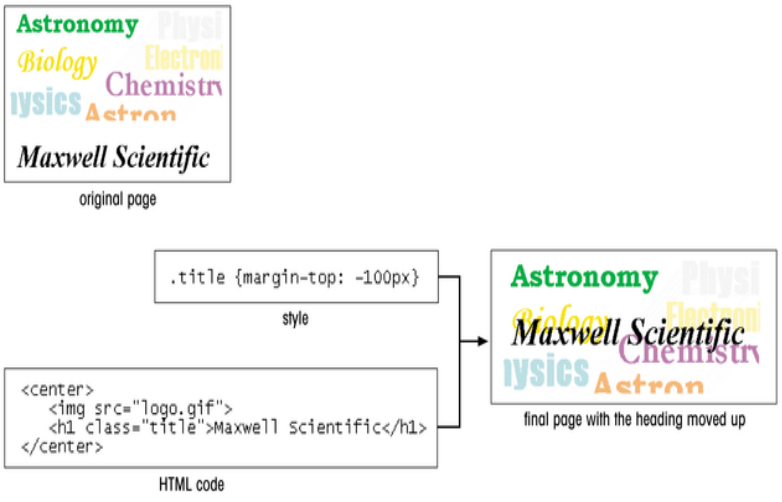
Controlling box margins

- The **margin** is the space between the block-level element and the parent element.
- four margin attributes:
  - **margin-top:** ***length*** - the space between the top of the box and the top margin
  - **margin-right:** ***length*** - the space between the right side of the box and the right margin
  - **margin-bottom:** ***length*** - the space between the bottom of the box and the bottom margin
  - **margin-left:** ***length*** - the space between the left side of the box and the left margin

Controlling box margins (cont.)

- margin sizes can be expressed in units of length (points, pixels, em units, etc.) or as a percentage of the width of the parent element box
- the value "**auto**" allows the browser to determine the margin size
- a margin size can be negative, although this can lead to unpredictable results when viewed with certain browsers.

Creating an overlay effect



Controlling box margins (cont.)

- The four **margin** attributes can be combined into a single attribute with the syntax:

  margin: *top right bottom left*

  e.g. #promoimage {float:right; margin:0em 0em 1em 1em}

  - if three values in the combined attribute, they are applied in order: top, right, bottom; the browser sets the left margin to match the right margin
  - if two values are specified, they are applied to the top and right margins; the browser sets the bottom and left margins to match the top and right margins
  - if only one value is specified, it is applied to all four margins

Setting box padding size

- four attributes are used to control the size of the element's padding:
  - **padding-top:** *length*
  - **padding-right:** *length*
  - **padding-bottom:** *length*
  - **padding-left:** *length*
- or use
  - **padding:** *top right bottom left*
- e.g., `h1{padding-top: 5ex}`

  `h2{padding: 5ex 2ex 5ex 2ex}`

Formatting box border

- three attributes for box border:

border-width: ***top right bottom left***

- border-top-width: *length*
- border-right-width: *length*
- border-bottom-width: *length*
- border-left-width: *length*

border-color: ***top right bottom left***

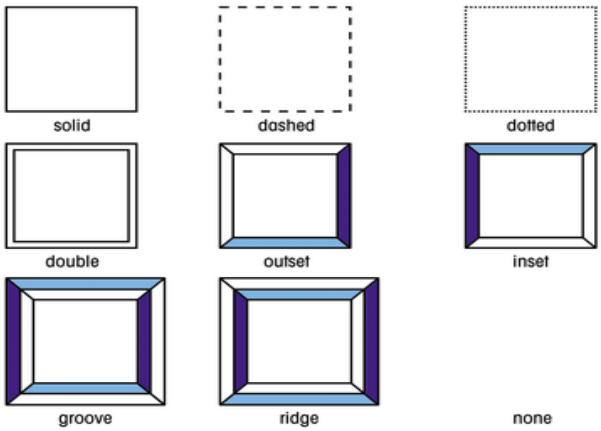
- border-top-color: *color*
- border-right-color: *color*
- border-bottom-color: *color*
- border-left-color: *color*

Formatting box border

border-style: ***top right bottom left***

- border-top-style: *type*
- border-right-style: *type*
- border-bottom-style: *type*
- border-left-style: *type*

Example of `border-style` values



Formatting box border (cont.)

- use **`border`** to combine the above three categories of box borders

  border: *width type color*

  or separately

  border-top: *width type color*

  border-right: *width type color*

  border-bottom: *width type color*

  border-left: *width type color*

- e.g.

```
address {border-top:0.5em double green}
h1 {border: 2px solid blue}
p {border-top-width: 4px; border-top-color: red}
```

Creating Rounded Corners

- Rounded corners can be applied to any of the four corners of a block element using the styles

`border-radius: top-left top-right bottom-right bottom-left;`

  - `border-top-left-radius: radius;`
  - `border-top-right-radius: radius;`
  - `border-bottom-right-radius: radius;`
  - `border-bottom-left-radius: radius;`

Formatting the width of block-level boxes

- use the **`width`** attribute
- box width can be expressed in terms of **absolute** or **relative** units of length, or as a **percentage** of the width of the parent element
  - example

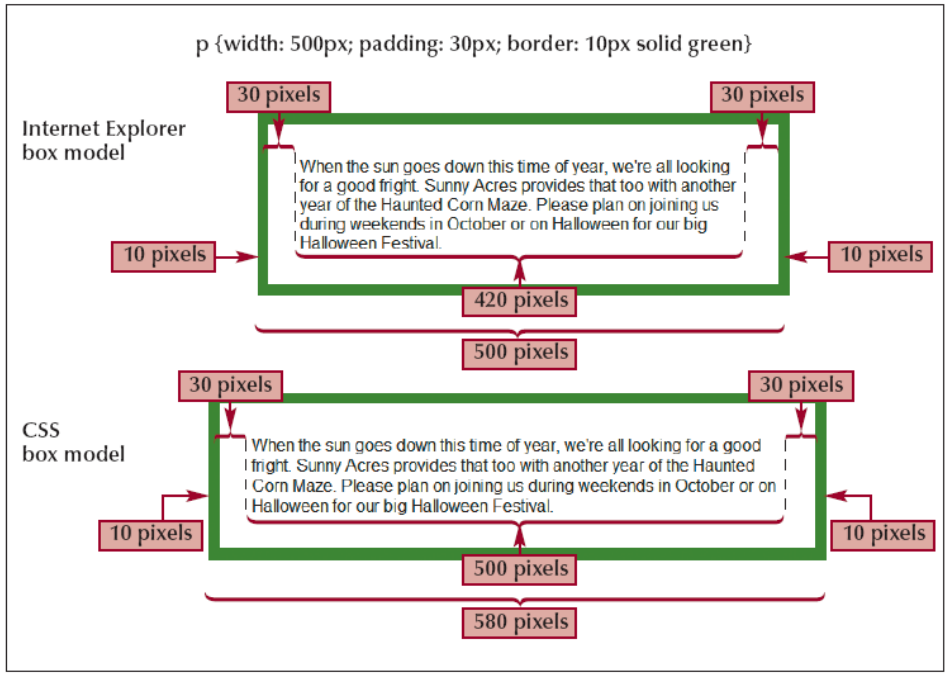
```
body {width: 75%}
```

  - reduces the width of the Web page body to 75% of the width of the browser's display area

Formatting the height of block-level boxes

- using the **height** attribute
- **height** can be expressed in absolute or relative lengths, but NOT percentages
- typically, the height of a block-level element is not set because problems can arise when the amount of text in the element exceeds the height allowed
- the **height** attribute is usually applied to inline images and little else
- e.g.,

```
p {width: 500px; padding: 30px; border: 10px
  solid green}
```

Managing Your Layout

- To add an outline around an element, use the style property

  `outline: width color style;`

  where *width*, *color*, and *style* are the outline width, outline color, and outline style, respectively

Working with container elements (revisit)

- HTML supports two types of container types:
  - the `<span>` tag, which is used to contain **inline elements** such as individual letters, words, phrases, or inline images
  - the `<div>` tag, which is used to group blocks of text such as paragraphs, block quotes, headings, or lists. Collectively, these text blocks are known as **block-level elements**

Positioning Objects

- To position an object at a specific coordinate, use the style properties

```
position: type;
top: value;
right: value;
bottom: value;
left: value;
```

where type indicates the type of positioning applied to the object (absolute, relative, static, fixed, or inherit), and the top, right, bottom, and left properties indicate the coordinates of the object

Positioning Objects

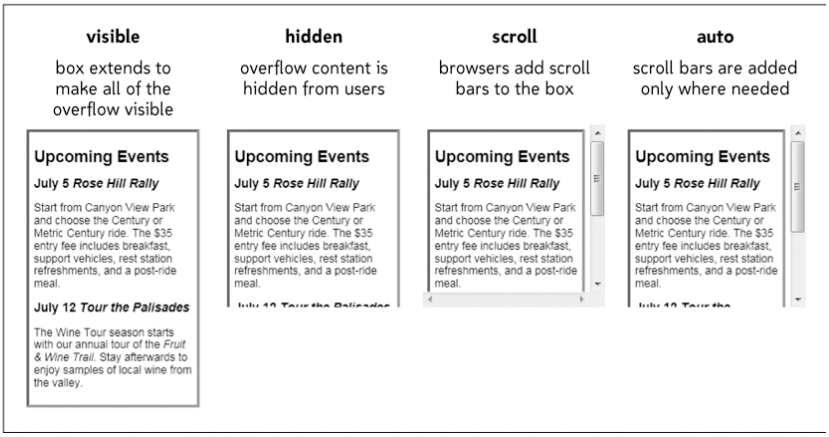
- **Absolute positioning** places an element at specific coordinates either in the page or within a container element
- **Relative positioning** is used to move an element relative to where the browser would have placed it if no positioning had been applied

Working with Overflow and Clipping

- When you force an element into a specified height and width, you can define how browsers should handle content that overflows allotted space using the style

```
overflow: type;
```


Working with Overflow and Clipping



Working with Overflow and Clipping

- To specify how browsers should handle content that overflows in the horizontal or vertical direction, use the following style:

```
overflow-x: type;
overflow-y: type;
```

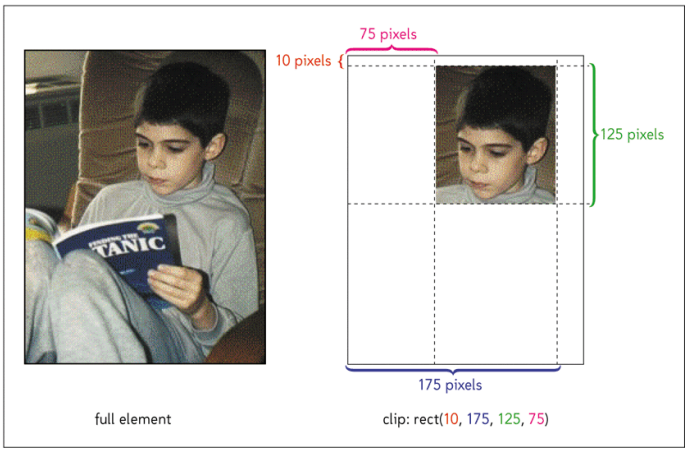

Working with Overflow and Clipping

- To clip an element's content, use the style

```
clip: rect(top, right, bottom, left);
```

where *top*, *right*, *bottom*, and *left* define the boundaries of the clipping rectangle

Clipping an Element



105

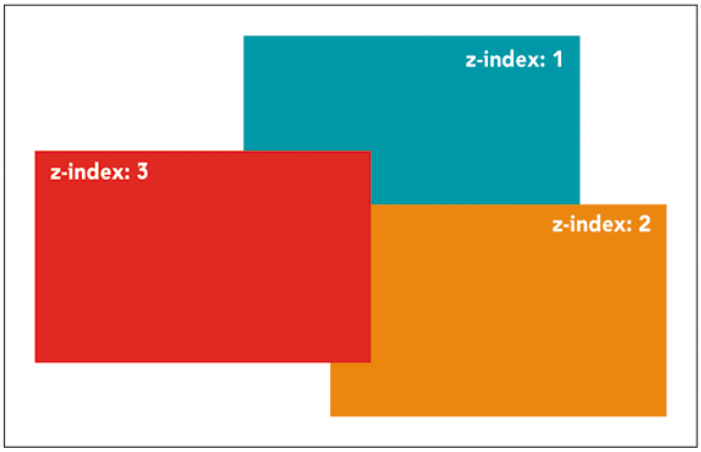
Stacking Elements

- Positioning elements can sometimes lead to objects that overlap each other
- By default, elements that are loaded later by the browser are displayed on top of elements that are loaded earlier
- To specify a different stacking order, use the style property

```
z-index: value;
```

106

Stacking Elements



Homework

- Finish reading tutorial 4
- Review assignment of Tutorial 4