1 **HTML5 and CSS3**

LAYOUT WITH CSS

2 **Styles and Layout**

- The default layout for web documents is a "flow" layout
  - Left-to right…top-to-bottom
- In the early days of HTML, the only tool available for more controlled page layout was tables
- CSS provides a more powerful tool for layout using new style properties for positioning elements exactly (or even relatively) on the page where the designer wants

3 **Flow Layout**

- Flow layout—elements appear in their natural location
- Displayed sequentially in order in which they are coded
- Each new element is left aligned unless otherwise stated
- Referred to as static for block elements
  - The default value for the position property

4 **Relative and Absolute Positioning**

- Modified positioning for block elements:
  - Relative positioning—"floating" an element's position with respect to its natural location, that is, relative to where it used to be within that flow
    - Relative to the position of the previous item in the HTML code
  - Absolute positioning—opposite of relative positioning in which elements are taken out of their natural flow and positioned absolutely with respect to their parent element
    - Either within browser window or some parent element

5 **Positioning an Object—Page 1**

- The position property is a box property which activates either relative or absolute positioning
- An element moved using absolute position affects the placement of elements that follow it

6 **Positioning an Object—Page 2**

- Format:
  - position: relative | absolute | fixed | initial | inherit | static;
    - relative—positioned relative its normal position (values of the top and left properties added to that position)
    - absolute—positioned relative to browser window (top and left represent elements exact location)
    - The bottom and right properties rarely are used

7 **Positioning an Object—Page 3**

- Examples:
  - img
    {
      position: relative;
    }
  - img
    {
      position: absolute;
      top: 100;
      left: 50
    }
  -

8 **Try It Out**

- Place the page header
- Place the sidebar
- Format the sidebar content
- Position the entire page body

9 **Working with Overflow—Page 1**

- Elements can be forced into a specific height and width using properties by those names
- Leaves the question what to do with the excess content
- CSS options are:
  - hidden—cuts off excess content
  - scroll—add horizontal and vertical scroll bars
  - auto—add scroll bars only as needed

10 **Working with Overflow—Page 2**

- Format:
  - overflow: visible | hidden | scroll | auto;
    - The default is visible which means expand the height to match the content
- Example
  - aside
    {
      height: 300px;
      overflow: auto;
    }

11 **Working with Overflow—Page 3**

- Overflow can be applied to just the horizontal or vertical property of an element

- Formats:
  - overflow-x: visible | hidden | scroll | auto;
  - overflow-y: visible | hidden | scroll | auto;

12 **Working with Overflow—Page 4**

- If the white-space property is set to the value nowrap, inline content is kept on a single line
- In combination with the overflow property with value set to auto, a vertical scroll bar is provided instead of a horizontal scroll bar, e.g.:
  - aside
    {
      width: 300px;
      overflow: auto;
      white-space: nowrap;
    }

13 **Working with Clipping**

- The clip property uses the rect() function to specify a rectangle to clip for an absolutely positioned element
- Anything outside the rectangle is hidden
- Often used in combination with overflow
- Format:
  - clip: rect(*top*, *right*, *bottom*, *left*);
- Example
  - aside
    {
      clip: rect(100, 150, 200, 250);
    }

14 **Try It Out**

- Define the overflow style for the aside element
- Style the page footer and the address

15 **Stacking Elements—Page 1**

- The z-index property is a "box" property which allows the developer to choose which elements appear on top of others if elements overlap, sort of 3-D positioning
- An element with a higher z-index property value will be higher on the stack of elements than one with a lower value
- Both positive and negative z-index values are allowed
- The z-index property only works on "positioned" elements
- Format:
  - z-index: *value*;

16 **Stacking Elements—Page 2**

- Example:
  - img.eagle
    {
      position: absolute;
      top: 100;
      left: 75;
      z-index: 0;
    }
    img.owl
    {
      position: relative;
      top: -70;
      left: 125;
      z-index: 1;
    }

17 **Columns—Page 1**

- CSS3 provides a feature for creating multiple text columns similar to newspaper columns
- The designer sets properties for:
  - The number of columns (this turns the feature "on")
  - The gap (sort of a margin) between columns
  - A rule (sort of a border between the columns)
- For older browers it may be necessary to use the browser specific prefixes, e.g. –moz, -webkit, -o, etc.

18 **Columns—Page 2**

- There are four properties related to this process:
  - column-count—the number of columns of text
  - column-gap—the width between the columns
  - column-rule—a border between the columns
    - Specify width, style and color similar to the other CSS border properties
  - column-span—used to have everything that preceded the element "span" all columns (not commonly supported except by Chrome and Safari)

19 **The column-count Property**

- Format:
  - *selector*
    {
      column-count: *numberOfColumns*;
    }

- Example:
  - article
    {
      -moz-column-count: 3;
      -webkit-column-count: 3;
      column-count: 3;
    }

20 **The column-gap Property**

- Format:
  - *selector*
    {
      column-gap: *size*;
    }
- Example:
  - article
    {
      -moz-column-gap: 25px;
      -webkit-column-gap: 25px;
      column-gap: 25px;
    }

21 **The column-rule Property**

- Format:
  - *selector*
    {
      column-rule: *width style color* ;
    }
- Example:
  - article
    {
      -moz-column-rule: 3px solid grey;
      -webkit-column-rule: 3px solid grey;
      column-rule: 3px solid grey;
    }