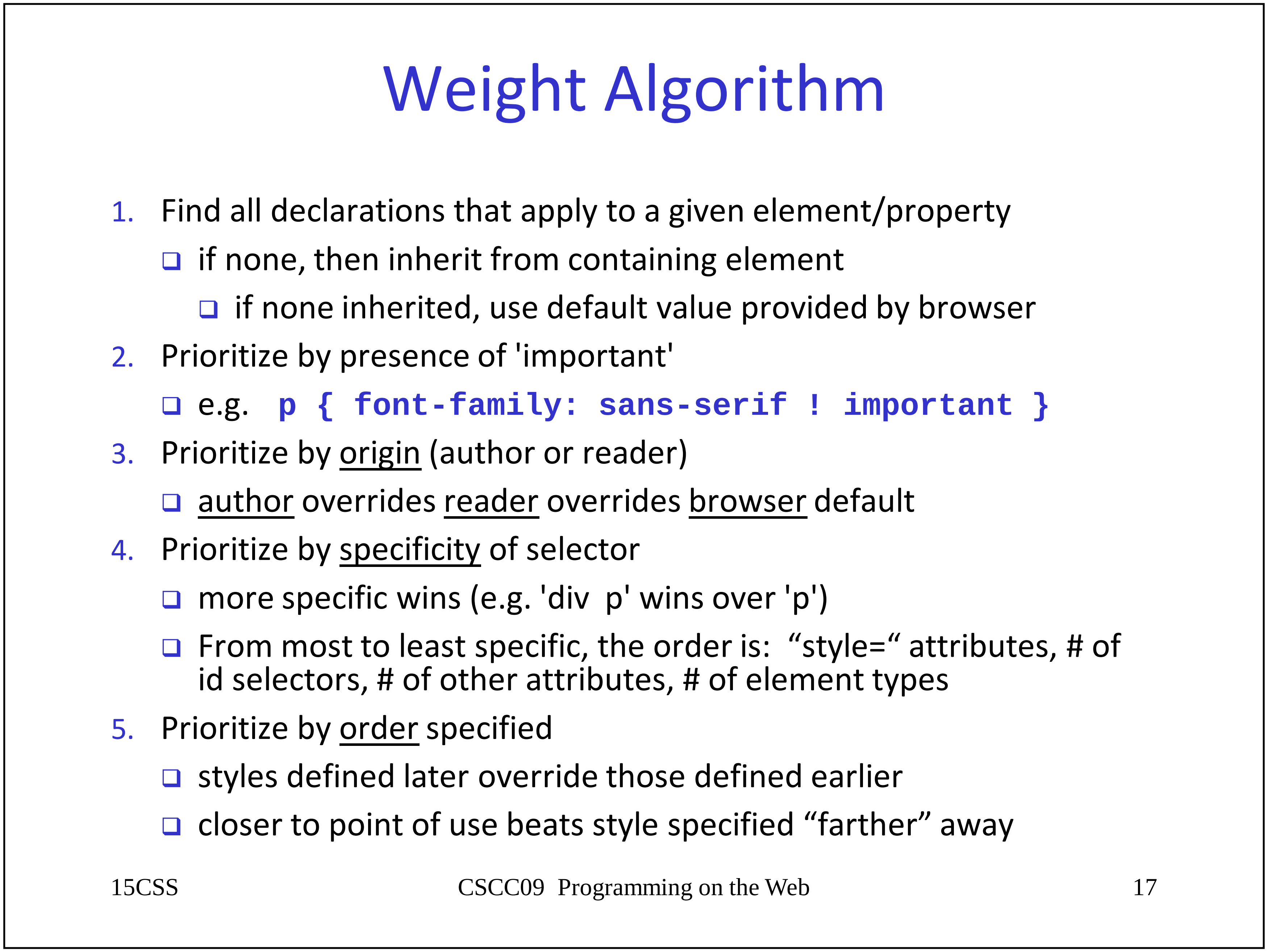
the Cascade

- >1 style sheet can simultaneously influence document styles
  - desirable for modularity of style design
  - style sheets can be defined for: entire Web site, specific Web document, individual user, browser preferences
- The CSS cascade assigns a weight to each style rule
- When several rules apply to a given element property, the one with the greatest weight takes precedence
- Generally behaves as you would hope, e.g. if you locally override one characteristic of an element's style, other characteristics are inherited unchanged



Weight Algorithm

1. Find all declarations that apply to a given element/property
   - if none, then inherit from containing element
     - if none inherited, use default value provided by browser
2. Prioritize by presence of 'important'
   - e.g. `p { font-family: sans-serif ! important }`
3. Prioritize by origin (author or reader)
   - author overrides reader overrides browser default
4. Prioritize by specificity of selector
   - more specific wins (e.g. 'div p' wins over 'p')
   - From most to least specific, the order is: “style=“ attributes, # of id selectors, # of other attributes, # of element types
5. Prioritize by order specified
   - styles defined later override those defined earlier
   - closer to point of use beats style specified “farther” away

CSS Style Properties

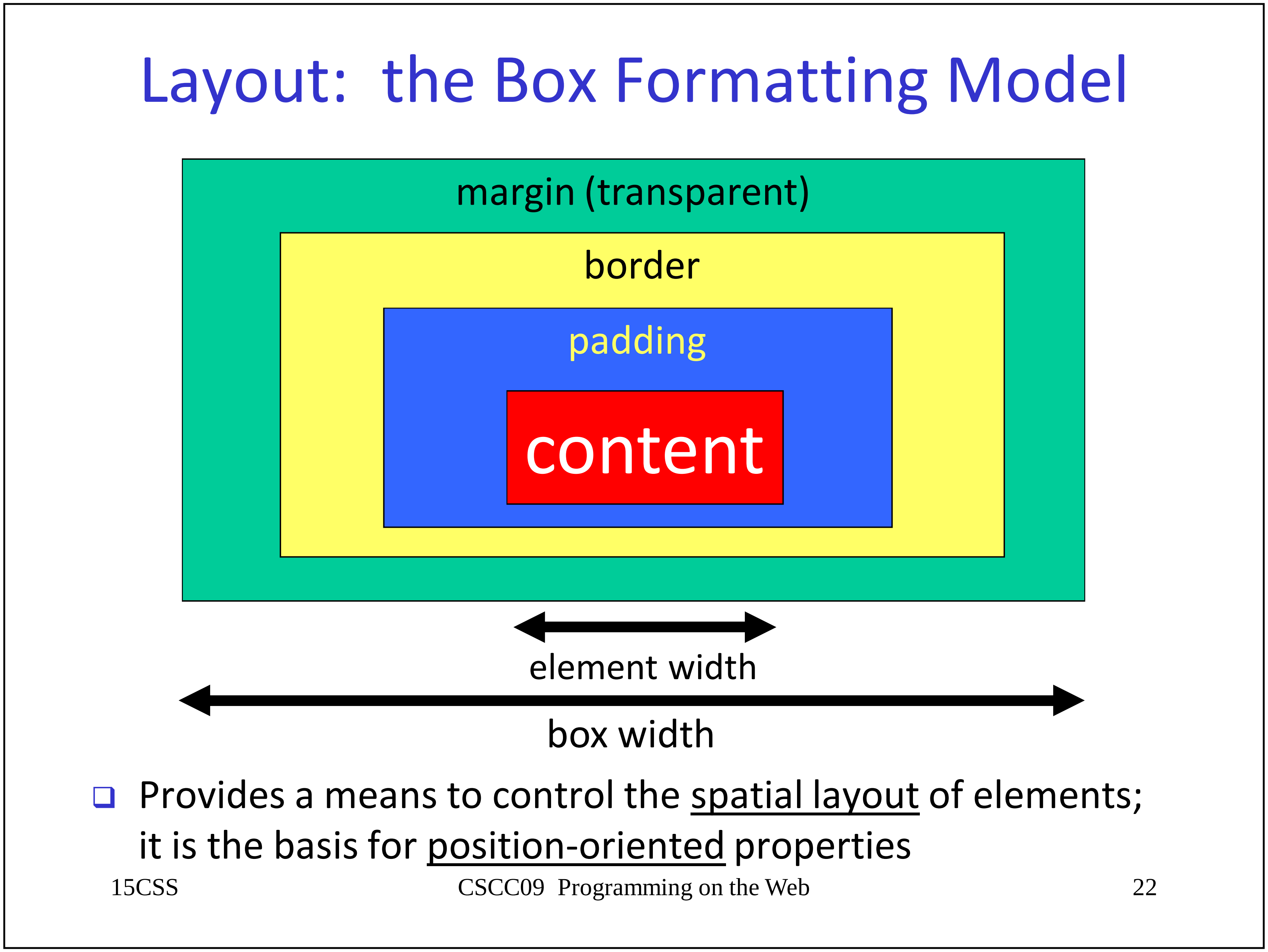
- CSS defines hundreds of style-properties for describing the nuances of document presentation, including: fonts; colors; background-colors and –images; padding, borders and margins; relative and absolute positioning; list properties; spacing; visibility, and many more …
- Here we cover only a tiny subset of these properties, but this subset includes some of the most commonly-used properties

Display Properties

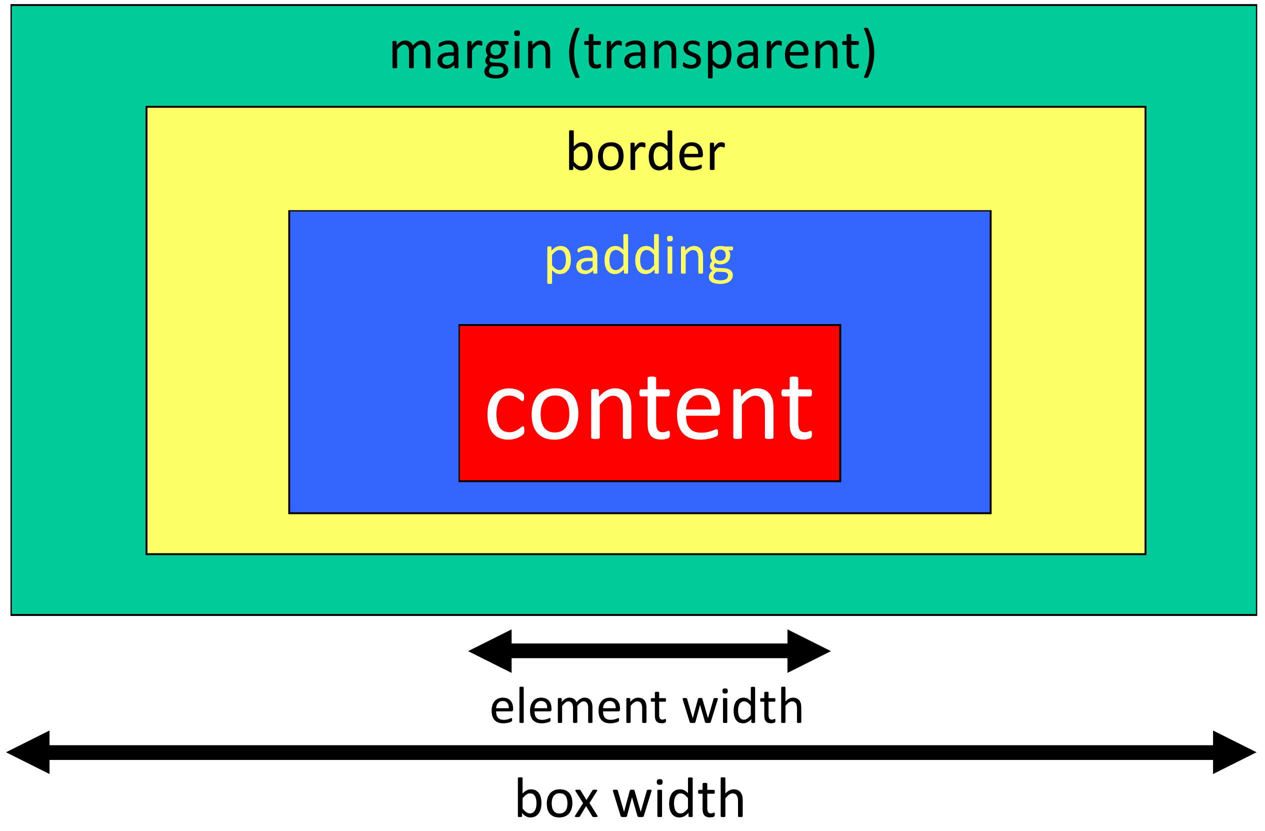
- `display:`
  - `display: block | list-item | inline | none |…`
    - block: opens a new box, with new-line before and after
    - list-item: same as block, with a list-item marker
    - inline: new box on same line as previous content (no line break before or after)
    - none: turns off display of the element (including children and surrounding box)
- e.g. HTML element display properties:
  - `p, ul { display: block }     // means what ?`
  - `em { display: inline }`
  - `li { display: list-item }`
  - `img { display: none }`

Visibility Properties

- **`visibility:`**
  - **`visibility: visible | hidden`**
  - default is visible
  - hidden can be used to show/hide dynamic HTML content
  - note, a hidden element takes up it's normal box-model browser-window space, whereas **`display:none`** does not
- **`opacity:`**
  - **`opacity: percentage-value (range 1.0 to 0.0)`**
  - percentage opaqueness (non-transparency) of element
  - **`e.g.  opacity:0.40;  /* this element is 60% transparent */`**



Layout: the Box Formatting Model



- Provides a means to control the spatial layout of elements; it is the basis for position-oriented properties

15CSS CSCC09 Programming on the Web 22

Box Properties

- margin-top, margin-right, margin-bottom, margin-left, margin, e.g. margin.html (transparent)
- border-top-width, border-right-width, border-bottom-width, border-left-width, e.g border.html
- padding-top, padding-right, padding-bottom, padding-left, padding, e.g. padding.html (inherits element background)

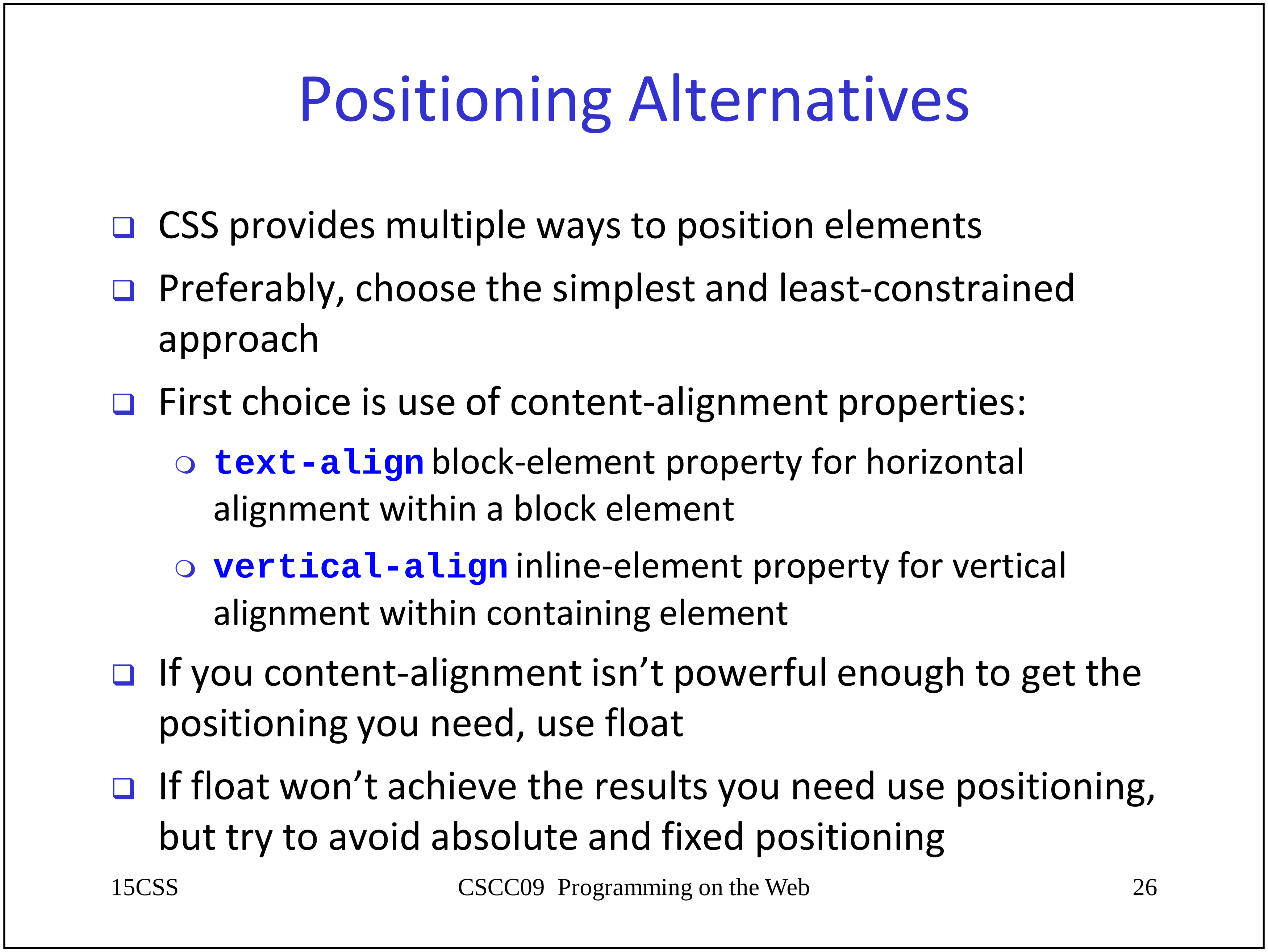
- margin, border, padding widths can be set using “compass-position”-ordered lists beginning with “north” or top, e.g.:
  margin: 1em 0 1em 1em;
  margin: 1em 0 1em; (left = right value)
  margin: 1em 0; (bottom/left = top/right)
  margin: 1em; (all values set the same)

Box-Content Properties

- Dimensions (block elements only):
  - width, e.g. “width: 400px”, “width: 70%”
  - max-width, min-width
  - height, e.g. “height 200px”, “height: 50%”
  - max-height, min-height
- Alignment (of and within block elements):
  - setting left and right margins to “auto” will center a block element – works best if element’s width is set
  - text-align: left | center | right (for inline elements within a block)

Positioning Properties

- position property
  - static: default
  - relative: offset from default static position
  - absolute: fixed position within containing element
  - fixed: fixed position within browser window
- box corner positions:
  - top, bottom, left, right
  - values: auto | *length* | initial | inherit
- e.g. `{ position:fixed; left:70px; top:20px; }`



Positioning Alternatives

- CSS provides multiple ways to position elements
- Preferably, choose the simplest and least-constrained approach
- First choice is use of content-alignment properties:
  - `text-align` block-element property for horizontal alignment within a block element
  - `vertical-align` inline-element property for vertical alignment within containing element
- If you content-alignment isn't powerful enough to get the positioning you need, use float
- If float won't achieve the results you need use positioning, but try to avoid absolute and fixed positioning

Float and Clear Properties

- **`float: left | right | none`**
  - floating element is removed from normal layout-flow
  - generally should specify a “width” attribute
  - left, right: formatted as block
    - left: moved to the left, text wraps on the right
    - right: moved to the right, text wraps on the left
- **`clear: none | left | right | both`**
  - list sides where floating elements are not accepted
    - e.g., if 'left‘, element will be moved below any left-floating element
- floatcss.html

Font Properties: font-family

- **`font-family: [family-name | generic-family] [, [family-name | generic-family]]*`**
  - priority order: left to right
  - e.g. (means what?):

```
body { font-family:gill,helvetica,sans-serif }
```

  - generic-families:
    - serif
    - sans-serif
    - cursive
    - fantasy
    - monospace
- **fontcss.html**

Font Properties

- **`font-weight: normal | bold | bolder | lighter | 100-900`**
  - always matches
- **`font-size:`** ***`absolute | relative | length | percentage`***
  - e.g. small, medium; larger, smaller; 10px, 1cm, 2em; 110%
- **`font-style: normal | italic | oblique`**
  - italic matches if italic or oblique found in known font list
  - else must match exactly
- **`font-variant: normal | small-caps`**
  - small-caps satisfied if font labeled as such or can be synthesized

Image Properties

- Images an important part of many Web sites
  - we've seen how to "float" them beside other content, and how to put borders, padding, and margins around them
  - what else might we want to do?
  - set as a page background:

    ```
    background-image: url("example.jpg")
    ```

  - a useful capability is setting of image opacity:

    ```
    img  { opacity:0.4; }
    img:hover { opacity:1.0; }
    ```

- **`bgd*.html`** (multiple examples)

15CSS CSCC09 Programming on the Web 31

Text Properties

- properties:
  - text-align: left | right | center | justify
  - text-shadow: *length*
  - word-spacing: normal | *length*
  - letter-spacing: normal | *length*
  - text-decoration: underline | overline | line-through
  - text-transform: capitalize | uppercase | lowercase | none
  - text-indent: *length* | *percentage*
  - line-height: *number* | *length* | *percentage*
- Example: `textcss.html`

Comments

- Just to keep things interesting, CSS comment syntax differs from that of HTML and JavaScript

```
/* this is a comment which can span one
   or more lines */
```

- Note that `//` and `<!--` do not work in CSS

Validation

- W3C provides a CSS validation service, that you can use to check your CSS for errors:
  - jigsaw.w3.org/css-validator/

- This validator is more picky about conformance to the CSS spec than Web browsers, which may render malformed CSS

Linking XML with CSS

1. Use the
   `<?xml-stylesheet type="text/css" … ?>`
   PI (Processing Instruction)
2. Use inline "style" attributes
   - Note, must declare these in the DTD

Example:

- resume_css.xml
- resume.css
- resume.dtd

Linking HTML with CSS

- Four methods for associating CSS definitions with HTML:
  1. **`<link rel="stylesheet" type="text/css" …>`** reference an external stylesheet, within the document **`<head>`** element
  2. @import an external stylesheet (rarely used)
  3. Embed a stylesheet definition within a **`<style>`** element, in the document **`<head>`** element
  4. Define inline styles using style-attributes: **`style="…"`**
- What are the dis/advantages of each?

Linking HTML with CSS

```
<html>
  <head>
      <title>CSS Linking Example</title>
      <link rel="stylesheet" type="text/css"
            href="http://www.utsc.utoronto.ca/style.css"/>
      <style type="text/css">
          @import url(http://www.utsc.utoronto.ca/basic.css);
          h1 { color: blue }      /* single style sheet with */
      </style>                    /* override on h1 color    */
  </head>
  <body>
    <h1>Headline is blue</h1>
    <p style="color: green">This paragraph is green.</p>
  </body>
</html>
```

Learning Objectives

- CSS advantages over XML/X-HTML element- and attribute-based styling
- Associating style with document elements: selectors
- Linking documents with CSS definitions
- Some basic style properties, including font families, sizes, display, the box-layout model, floating elements