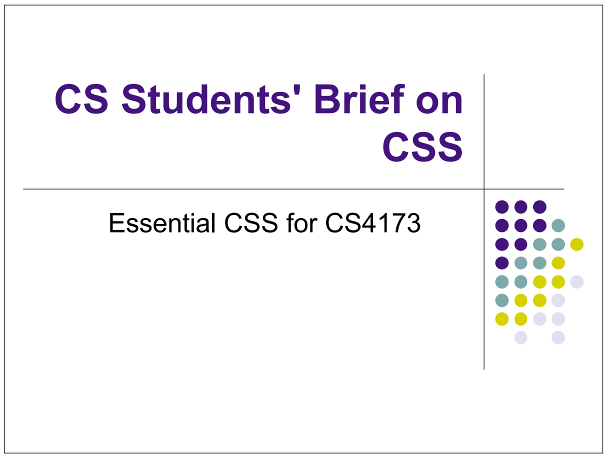
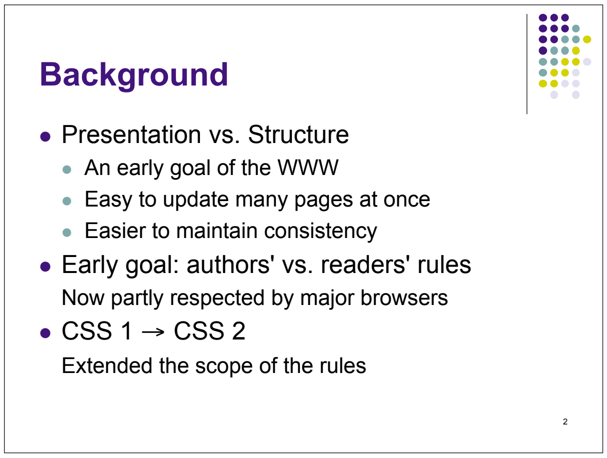


CS Students' Brief on CSS

Essential CSS for CS4173



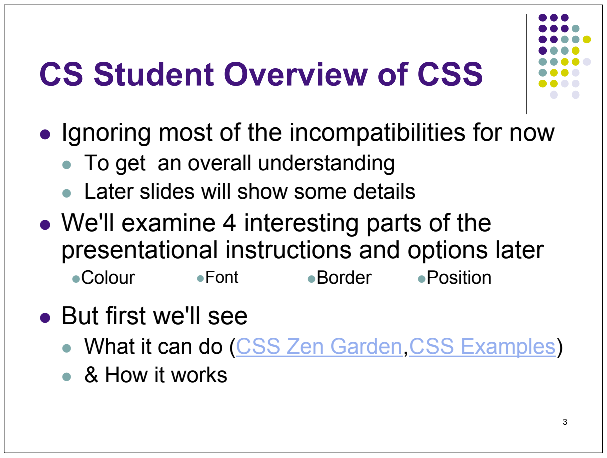
Background

- Presentation vs. Structure
  - An early goal of the WWW
  - Easy to update many pages at once
  - Easier to maintain consistency
- Early goal: authors' vs. readers' rules

  Now partly respected by major browsers
- CSS 1 → CSS 2

  Extended the scope of the rules

2



CS Student Overview of CSS

- Ignoring most of the incompatibilities for now
  - To get an overall understanding
  - Later slides will show some details
- We'll examine 4 interesting parts of the presentational instructions and options later
  - Colour
  - Font
  - Border
  - Position
- But first we'll see
  - What it can do (CSS Zen Garden,CSS Examples)
  - & How it works

3

What's Next?

- Introduction to CSS rule method
- CSS selectors
- How CSS matches rules to elements
  - The parse tree
  - The cascade
- How to include rules in an XHTML file
  - A simple example
- Visual formatting and Dual presentation

4

How CSS Works — Rules

- Rules provide *presentation hints* to browser
  - Browser can ignore hints
  - Three sources of rules:
    - User agent (browser's default settings),
    - Webpage (source file),
    - The user (personal settings in the browser)
- Rules apply when *selectors* match context
  - E.g. `p {text-indent:1.5em }`
  - Selector is `p` (matches any `<p>` element)

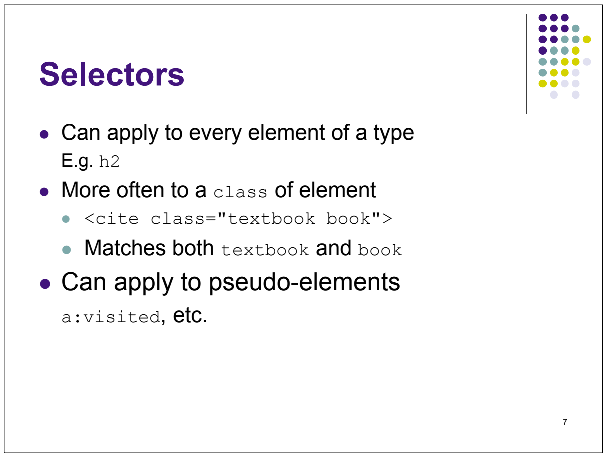
5

Rules

- Attached to elements
  - As attributes of elements (inline style)
  - Tied to `id` attribute of elements
  - Tied to `class` attribute of elements
- Rules all have form

  `{Property Name : Value;}`
- Multiple rules separated by `;`

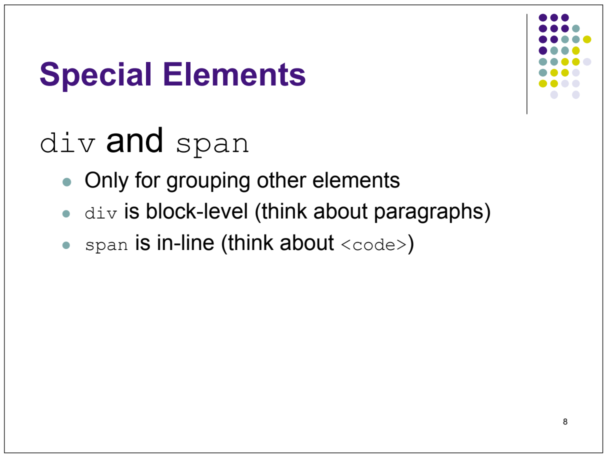
6



Selectors

- Can apply to every element of a type
  E.g. `h2`
- More often to a `class` of element
  - `<cite class="textbook book">`
  - Matches both `textbook` and `book`
- Can apply to pseudo-elements
  `a:visited`, etc.

7

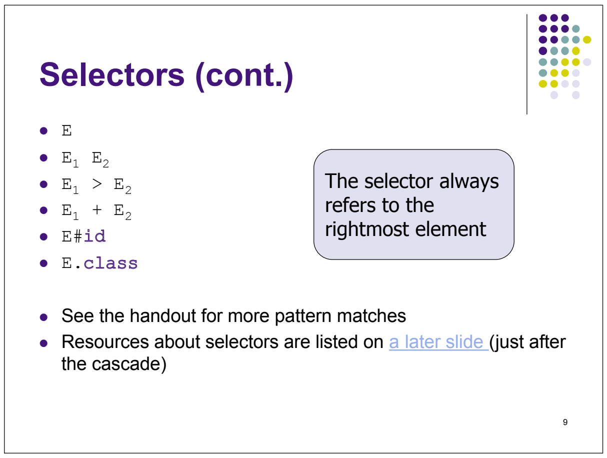


Special Elements

`div` and `span`

- Only for grouping other elements
- `div` is block-level (think about paragraphs)
- `span` is in-line (think about `<code>`)

8



Selectors (cont.)

- E
- $E_1\ E_2$
- $E_1 > E_2$
- $E_1 + E_2$
- `E#id`
- `E.class`

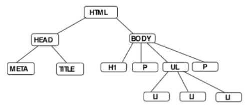
The selector always refers to the rightmost element

- See the handout for more pattern matches
- Resources about selectors are listed on a later slide (just after the cascade)

9

How CSS Works — Matching

- Every XHTML document represents a *document tree*

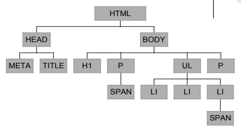


- The browser uses the tree to determine which rules apply
- What about inheritance? And conflicts?

10

HTML Parse Tree

```
<html>
   <head>
      <meta … />
      <title>…</title>
   </head>
   <body>
      <h1>…</h1>
      <p>…<span>…</span>…</p>
      <ul>
         <li>…</li>
         <li>…</li>
         <li>…<span>…</span>…</li>
      </ul>
      <p>…</p>
   </body>
</html>
```



- What will `h1 + p` match?
- What will `ul > span` match?
- What will `ul {color:blue}` do?

13

Inheritance in CSS ⇒ The Cascade

- Inheritance moves down tree
- Cascading move horizontally
  - It works on elements that the same rules apply to
  - It is only used for tie-breaking when ≥2 rules apply
- The highest ranking rule wins
- Most specific wins (usually)
- But important rules override others
  - `!important` beats plain
  - User's `!important` beats everything else
- See the specificity section of the CSS standard!

14

Details of the CSS2.1 Cascade

For each element **E**

1. Find all declarations that apply to **E**
2. Rank those declarations by origin
   a. user `!important` > author `!important` > inline style
   b. inline style > author plain > user plain > browser
3. If there is not a clear winner then most specific rule wins.
   Compute specificity as shown on next slide.

15

CSS2.1 Cascade (Continued)

3. Compute specificity thus:
   a. If one rule uses more `#` symbols than the others then it applies, otherwise …
   b. If one rule uses more attributes and pseudo-elements than the others then it applies, otherwise …
   c. If one rule uses more real (not pseudo) elements then it applies
   d. For each two rules that have the same number of every one of the above specifiers, the one that was declared last applies

- An equivalent method is shown on the next slide

16

CSS 2.1 Cascade Computation

- The cascade algorithm in the standard uses a semi-numerical algorithm
- The computation looks like this:

$$a = \begin{cases} 1 & \text{if the selector is an inline style} \\ 0 & \text{otherwise} \end{cases}$$

b = Number of `id` attributes (but only if specified with `#`)

c = Number of attributes (except those in b) and pseudo-elements specified

d = Number of plain (non-pseudo) and non-`id` elements specified

- The specificity is $a \times base^3 + b \times base^2 + c \times base + d$
  - Where base = 1 + maximum(b,c,d)
  - The rule with the largest specificity applies

17

Selector Resources on the WWW

- The CSS 2 Standard
  - At W3.org (http://www.w3.org/TR/REC-CSS2/)
  - In frames (http://www.meyerweb.com/eric/css/references/css2ref.html)
- Selector Tutorial [Excellent!] (http://css.maxdesign.com.au/selectutorial/)
- SelectORACLE (http://gallery.theopalgroup.com/selectoracle/)
- Other Recommended Resources
  - In the resources part of the course website

18

How To Include Rules

- Inline
  - `<p style="text-align: center" >…</p>`
- Inside the `head` element
  - `<link rel="stylesheet" type="text/css" href="site.css" />`
  - `<style type="text/css">…</style>`

  -
    ```
    <style type="text/css">
      @import url(site.css);
      /* other rules could go here */
    </style>
    ```

19

Simple Example

- Fonts and background colours
- Inheritance and cascading

  - See simple in CSS examples

20

A Very Brief Overview of Visual Formatting With CSS

- Visual Formatting
  - Fonts
  - Colours
  - Position
  - Box model and Borders
- Dual presentation / Hiding CSS

21

Visual Formatting: fonts

- Some major properties
  - `font-family`
    - `body {font-family: Garamond, Times, serif}`
    - Serif fonts and **sans-serif** fonts
  - `font-size`:
    Length (em,ex), percentage, relative size, absolute size
  - `font-style:`
    Normal, italic, oblique
  - `font-weight:`
    Lighter, normal, bold, bolder, 100, 200, …, 800, 900
- Set all at once with `font`

22

Visual Formatting: Colours

- How to specify
  - 16 Predefined names
  - RGB values (%, #, 0…255)
  - System names: e.g. `CaptionText`
- Dithered Colour
  - See Lynda Weinman's charts
  - Okay for photos, etc.

23

Visual Formatting: Colours (cont.)

- Major properties
  - `background-color`
  - `color`
- `transparent` and `inherit` values

24

Visual Formatting: Images

- `position`:
  `static`, `relative`, `absolute`, `fixed`
- Static — normal elements
- Relative — translate from usual position
- Absolute — scroll with the page
- Fixed — like absolute, but don't scroll away
- Example: Jon Gunderson

25

Visual Formatting: Images (cont.)

- `z-index`: depth
- `float` and `clear`
  - `float: left` or `float: right` or `float: none`
    Position relative to parent element
  - Reset with clear
    `<br style="clear:both" />`

26

Visual Formatting: Box Model

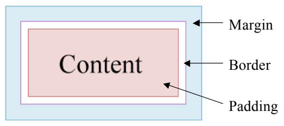


Figure from materials © by Dietel, Dietel, and Nieto

Box Model (Cont.)

- Padding
  - Size in %, em, or ex for text
  - `padding-top`, `padding-right`, `padding-bottom`, `padding-left`
    Mnemonic: TRouBLe
  - Set all at once with `padding`
- Margin
  - Similar to padding
  - But can also be `auto`
    see centring example

Borders? Do we have borders!

- Four types again
- Can all be set at once with `border`
- See Border slides by Jon Gunderson

CSS For Dual Presentation

- What if users don't have CSS?
  See button example
- What if CSS only sortof works?
  Tricks to hide CSS from dumb browsers
- How can I make cool webpages?
  One of many ways: see W3C Core Styles

30

Hiding CSS — Why do we need to?

- Two failure modes: graceful and catastrophic
- Pragmatism
- Hubris

31

A Trick For Dual Presentation

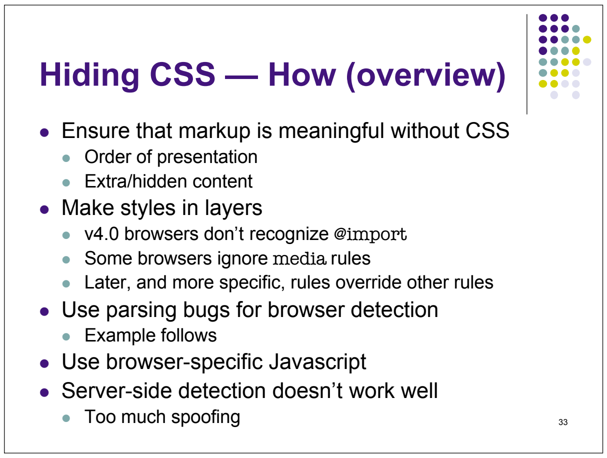
- `visibility`: `visible` or `hidden`
- `display`: `none`

`visible:hidden`
element can't be seen but it still uses space

`display:none`
element isn't shown

`visibility` example (CSS buttons)

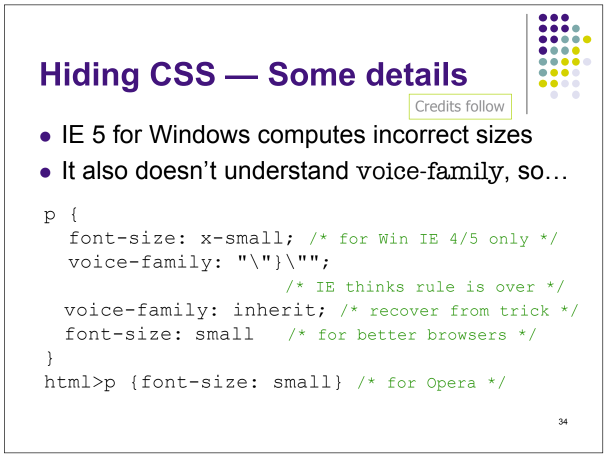
32



Hiding CSS — How (overview)

- Ensure that markup is meaningful without CSS
  - Order of presentation
  - Extra/hidden content
- Make styles in layers
  - v4.0 browsers don't recognize `@import`
  - Some browsers ignore `media` rules
  - Later, and more specific, rules override other rules
- Use parsing bugs for browser detection
  - Example follows
- Use browser-specific Javascript
- Server-side detection doesn't work well
  - Too much spoofing

33



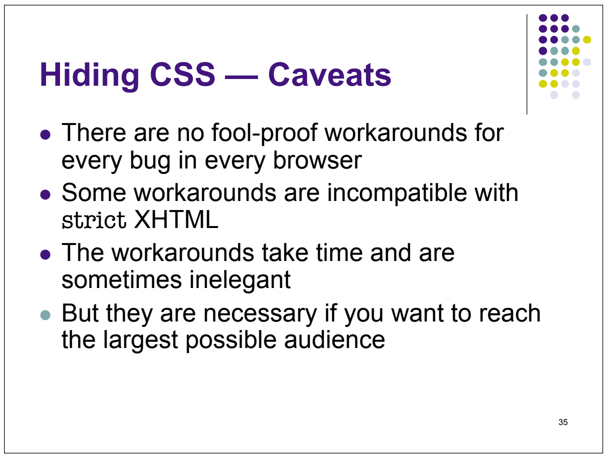
Hiding CSS — Some details

Credits follow

- IE 5 for Windows computes incorrect sizes
- It also doesn't understand `voice-family`, so…

```
p {
   font-size: x-small; /* for Win IE 4/5 only */
   voice-family: "\"}\"";
                        /* IE thinks rule is over */
  voice-family: inherit; /* recover from trick */
  font-size: small    /* for better browsers */
}
html>p {font-size: small} /* for Opera */
```

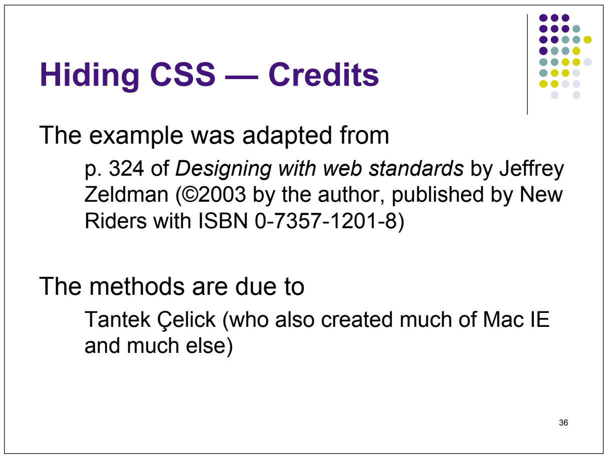
34



Hiding CSS — Caveats

- There are no fool-proof workarounds for every bug in every browser
- Some workarounds are incompatible with `strict` XHTML
- The workarounds take time and are sometimes inelegant
- But they are necessary if you want to reach the largest possible audience

35



Hiding CSS — Credits

The example was adapted from

p. 324 of *Designing with web standards* by Jeffrey Zeldman (©2003 by the author, published by New Riders with ISBN 0-7357-1201-8)

The methods are due to

Tantek Çelick (who also created much of Mac IE and much else)

36