


CS Students' Brief on CSS


Essential CSS for CS3172



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`
- `E1 E2`
- `E1 > E2`
- `E1 + E2`
- `E#id`
- `E.class`

The selector always refers to the rightmost element

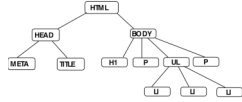
- See the handout for more pattern matches
- Resources about selectors are listed on [a later slide](#) (just after the cascade)



9

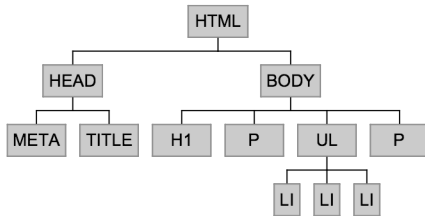
How CSS Works — Matching

- Every XHTML document represents a *document tree*



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

10

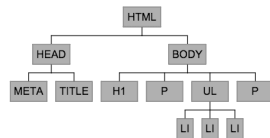


11

HTML Parse Tree

```

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



12

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>

```

```

graph TD
    HTML[HTML] --> HEAD[HEAD]
    HTML --> BODY[BODY]
    HEAD --> META[META]
    HEAD --> TITLE[TITLE]
    HEAD --> H1[H1]
    BODY --> P1[P]
    BODY --> UL[UL]
    BODY --> P2[P]
    P1 --> SPAN1[SPAN]
    UL --> LI1[LI]
    UL --> LI2[LI]
    UL --> LI3[LI]
    LI3 --> SPAN2[SPAN]

```

- 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

14

Details of the CSS 2.1 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 2 slides.

15

CSS 2.1 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 (including `class`) than the others then it applies, otherwise ...
 - c. If one rule uses more 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
 - `class` is the only attribute that can be selected with the `.` in CSS
 - 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:

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

class is an attribute
- $b =$ Number of `id` attributes (but only if specified with #)
- $c =$ Number of attributes (except those in b) and pseudo-attributes specified
- $d =$ Number of non-`id` elements specified (including pseudo-elements)
- The specificity is $a \times \text{base}^3 + b \times \text{base}^2 + c \times \text{base} + d$
 - Where $\text{base} = 1 + \text{maximum}(b, c, d)$
 - The rule with the largest specificity applies

17

To find the value for an element/property combination, user agents must apply the following sorting order:

1. Find all declarations that apply to the element and property in question, for the target media type. Declarations apply if the associated selector matches the element in question.
2. Sort according to importance (normal or important) and origin (author, user, or user agent). In ascending order of precedence:
 - a. user agent declarations
 - b. user normal declarations
 - c. author normal declarations
 - d. author important declarations
 - e. user important declarations
3. Sort rules with the same importance and origin by specificity of selector: more specific selectors will override more general ones. Pseudo-elements and pseudo-classes are counted as normal elements and classes, respectively.
4. Finally, sort by order specified: if two declarations have the same weight, origin and specificity, the latter specified wins. Declarations in imported style sheets are considered to be before any declarations in the style sheet itself.


Apart from the 'important' setting on individual declarations, this strategy gives author's style sheets higher weight than those of the reader. User agents must give the user the ability to turn off the influence of specific author style sheets, e.g., through a pull-down menu.

CSS 2.1 §6.4.1 Cascading order

Summary
CSS 2.1 Cascade:

19

Pseudo-Elements? Pseudo-Attributes?!




'CSS introduces the concepts of pseudo-elements and pseudo-classes to permit formatting based on information that lies outside the document tree.'

- **Classes**
 - :first-child
 - :link, :visited
 - :hover, :active, :focus
 - :lang
- **Elements**
 - :first-line
 - :first-letter
 - :before, :after

CSS 2.1 §5.10
Pseudo-elements and pseudo-classes

20


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](#)

21

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>`

22

Simple Example

- Fonts and background colours
- Inheritance and cascading


➤ See [simple](#) in CSS examples



23

A Very Brief Overview of Visual Formatting With CSS


- Visual Formatting
 - [Fonts](#)
 - [Colours](#)
 - [Position](#)
 - [Box model and Borders](#)
- [Dual presentation / Hiding CSS](#)



24

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`



25

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.



26

Visual Formatting: Colours (cont.)


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



27

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](#)



28

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" />`

29

Visual Formatting: Box Model

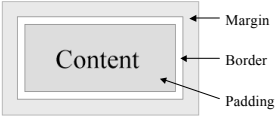


Figure from materials © by Dietel, Dietel, and Nieto

30

Borders? Do we have borders!

- Four types again
- Can all be set at once with `border`
- See [Border slides](#) by Jon Gunderson

31

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

Width is of content only.
Neither the border nor the padding are included in width.

32

Making Room for a fixed position object

CS4172 > Drs

Web-centric Computing

Course Administrivia

[Course | Announcements | Materials | Resources] [Course sitemap]

Syllabus
Basic syllabus (rules and dates)
Addendum to syllabus (topics)
both in PDF format

Goals and Topics for the course
in a separate document

Lecture schedule

```
body
{margin-left: 6.3em}
div.up
{position: fixed;
left: 1em;
top: 40%;
padding: .2ex;
min-width: 5.5ex }
```

Width computation: see <URL:
<http://tantek.com/CSS/Examples/boxmodelhack.html>>

33

Formatting The 'Jump Box'

'Jump Box'

CS4172 > Drs

Web-centric Computing

Course Administrivia

[Course | Announcements | Materials | Resources] [Course sitemap]

Syllabus
Basic syllabus (rules and dates)
Addendum to syllabus (topics)
both in PDF format

Goals and Topics for the course
in a separate document

Lecture schedule

34

Basic Formatting of the 'Jump Box'

HTML Outline

```
<body>
<!-- ... -->
<div class="up">
  <dl>
    <dt>Jump to
      top</dt>
  <!-- ... -->
</div>
</body>
```

Extract of CSS Rules

```
body
{margin-left: 6.3em}
div.up
{position: fixed;
left: 1em;
top: 40%;
padding: .2ex;
min-width: 5.5ex }
```

35

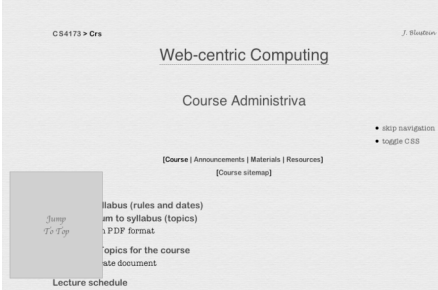
body {padding: 4em}

37

div.up {margin: 4em}

38

div.up dl {margin:4em}



CS4172 > Cms J. Blodgett

Web-centric Computing

Course Administriva

[Course | Announcements | Materials | Resources]

[Course sitemap]

- HELP navigation
- toggle CSS

Jump To Top

llabus (rules and dates)
 an to syllabus (topics)
) PDF format
 topics for the course
 site document

Lecture schedule

39

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](#)

40

**Hiding CSS —
Why do we need to?**

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

41

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)

42

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

43

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 */
    
```

44

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

- For more about hacks see
<URL:<http://tantek.com/log/2005/11.html>>

45

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)

46
