

Cascading Style Sheets (CSS)

Structure vs. Presentation

- HTML Original Intent:
 - Author: specifies document structure, not presentation
 - Browser: renders marked-up content into human-readable output
 - Formatting (layout) based on its environment: desktop, laptop, palmtop, web phone, etc.
- In Practice:
 - Designers wanted much more control over layout
 - Browsers added growing number of ad hoc tags and attributes
 - Browser-specific extensions are a nightmare for portability
 - HTML 'degenerated' into a markup language for format
 - HTML was not designed for this purpose

2,400 HTML characters to describe 60 characters of content

Emp	Salary	Age	Sex
John	10000	35	M
Jane	12000	30	F
Bob	15000	40	M
Alice	18000	25	F

```

<table border="1">
  <thead>
    <tr>
      <th>Emp</th>
      <th>Salary</th>
      <th>Age</th>
      <th>Sex</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>10000</td>
      <td>35</td>
      <td>M</td>
    </tr>
    <tr>
      <td>Jane</td>
      <td>12000</td>
      <td>30</td>
      <td>F</td>
    </tr>
    <tr>
      <td>Bob</td>
      <td>15000</td>
      <td>40</td>
      <td>M</td>
    </tr>
    <tr>
      <td>Alice</td>
      <td>18000</td>
      <td>25</td>
      <td>F</td>
    </tr>
  </tbody>
</table>

```

```

<table border="1">
  <thead>
    <tr>
      <th>Emp</th>
      <th>Salary</th>
      <th>Age</th>
      <th>Sex</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>10000</td>
      <td>35</td>
      <td>M</td>
    </tr>
    <tr>
      <td>Jane</td>
      <td>12000</td>
      <td>30</td>
      <td>F</td>
    </tr>
    <tr>
      <td>Bob</td>
      <td>15000</td>
      <td>40</td>
      <td>M</td>
    </tr>
    <tr>
      <td>Alice</td>
      <td>18000</td>
      <td>25</td>
      <td>F</td>
    </tr>
  </tbody>
</table>

```

Problems

Cascading Style Sheets (CSS)

- Separate structure from presentation
- “Simple” mechanism to attach style to structured documents
 - fonts, colours, spacing, ...
- Applies generically to all forms of XML
- Removes the requirement for further formatting tags from HTML proper

CSS Advantages

- Precise control over presentation
- Simplify site maintenance
- Faster downloads
- Resolution independence
- Media-specific rendering

CSS Language

stylesheet: ruleset*

ruleset: selector '{' [declaration ';']* '}'

declaration: property ':' expr ['! important']?

```
selector {
  property: value;
  property: value;
  ...
  property: value;
}

p {
  font-family: sans-serif;
  color: red;
}
```

Attaching a CSS file: <link>

```
<head>
  ...
  <link href="filename" type="text/css" rel="stylesheet" />
  ...
</head>
<link href="style.css" type="text/css" rel="stylesheet" />
<link href="http://www.google.com/uds/css/gsearch.css"
  rel="stylesheet" type="text/css" />
```

- A page can link to multiple style sheet files
- In case of a conflict (two sheets define a style for the same HTML element), the latter sheet's properties will be used

Embedding style sheets: <style>

```
<head>
  <style type="text/css">
    p { font-family: sans-serif; color: red; }
    h2 { background-color: yellow; }
  </style>
</head>
```

- CSS code can be embedded within the head of an HTML page
- this is *bad style* and should be avoided when possible (why?)

Inline styles: the style attribute

```
<p style="font-family: sans-serif; color: red;">
This is a paragraph</p>
```

- higher precedence than embedded or linked styles
- used for one-time overrides and styling a particular element
- this is *really bad style* and should be avoided when possible (why?)

Selectors

- Type E
- Universal *
- Grouping E,G,F
- Attribute E[foo="hi"]
- ID #myID or E#myID
- Class .myClass
- Pseudo-element E:pseudo-element
- Contextual
 - Descendent E F
 - Child E > F
 - Adjacent E + F

Element Selector

```
h1 {
  border: solid red 1;
  margin: 1cm;
  padding: 1em;
  width: 40%;
  text-align: left;
}
```

Grouping

- One can group multiple headings, and apply a set of styles to all.

```
h1, h3 {
  color: blue;
  font-family: helvetica
}
```

13

Attribute as Selector

- Use attribute values to conditionally apply style

- XHTML
 - `<li class="grad">M.Sc.`
 - `<li class="undergrad">B.Sc.`

- CSS
 - `li[class="undergrad"] { color: black;}`
 - `li[type="grad"] { color: blue;}`

14

ID as Selector

- XHTML allows any element to have a special "id" attribute that is unique in the document.

- Can be used as the target for a hyperlink
- Can be used to associate style properties with a particular element
- XHTML
 - `<li id="i1">Programming the WWW`

- CSS
 - `li#i1 { color: blue;}`
 - `li[id="i1"]`
 - `#i1`

15

Class as Selector

- Use class to conditionally apply style

- XHTML
 - `<tr class="header"> ...`

- CSS
 - `tr.header { color: blue;}`
 - `.header { color: blue;}`

16

Multiple class selector example

```
<h2>Spatula City! Spatula City!</h2>
<p class="special">See our spectacular spatula specials!</p>
<p class="special standout">Satisfaction guaranteed.</p>
<p class="standout">We'll beat any advertised price!</p>
```

```
.special {
  background-color: yellow;
  font-weight: bold;
}
p.standout {
  color: red;
  font-family: cursive;
}
```

css

Spatula City! Spatula City!

See our spectacular spatula specials!

Satisfaction guaranteed.

We'll beat any advertised price!

output

pseudo-classes

class	description
:active	an activated or selected element
:focus	an element that has the keyboard focus
:hover	an element that has the mouse over it
:link	a link that has not been visited
:visited	a link that has already been visited
:first-letter	the first letter of text inside an element
:first-line	the first line of text inside an element
:first-child	an element that is the first one to appear inside another

```
a:link { color: #FF0000; } /* unvisited link */
a:visited { color: #00FF00; } /* visited link */
a:hover { color: #FF00FF; } /* mouse over link */
```

Context selectors

```
selector1 selector2 {
  properties
}
```

- applies the given properties to selector2 only if it is inside a selector1 on the page

```
selector1 > selector2 {
  properties
}
```

- applies the given properties to selector2 only if it is directly inside a selector1 on the page (selector1 tag is immediately inside selector2 with no tags in between)

Context selector example

```
<p>Shop at <strong>Hardwick's Hardware</strong>...</p>
<ul>
  <li>The <strong>best</strong> prices in town!</li>
  <li>Act while supplies last!</li>
</ul>
#ad li.important strong { text-decoration: underline; }
```

Shop at **Hardwick's Hardware...**

- The **best** prices in town!
- Act while supplies last!

Context selector example 2

```
<div id="ad">
  <p>Shop at <strong>Hardwick's Hardware</strong>...</p>
  <ul>
    <li class="important">The <strong>best</strong>
      prices in town!</li>
    <li>Act <strong>while supplies last!</strong></li>
  </ul>
</div>
#ad li.important strong { text-decoration: underline; }
```

Shop at **Hardwick's Hardware...**

- The **best** prices in town!
- Act **while supplies last!**

Inheritance

- Most styles are inherited into nested elements.
- Can set a "default" style by setting property values for the <body> element.

```
body {
  font-size: 16px;
}
```

The Cascade

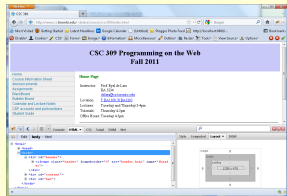
- It's called Cascading Style Sheets because the properties of an element *cascade* together in this order:
 - browser's default styles
 - external style sheet files (in a <link> tag)
 - internal style sheets (inside a <style> tag in the page's header)
 - inline style (the style attribute of the HTML element)
- For different properties, all matching selectors are applied
- In the case of a property conflict
 - each rule is assigned a weight
 - heaviest wins

Weight Algorithm

1. Find all declarations that apply to the element/property
 - if none, then inherit
 - if no inheritance, use initial value
2. Sort by presence of '!important'
3. Sort by origin (author,user,browser)
 - author wins over user, user wins over browser default
4. Sort by specificity of selector
 - more specific wins ('h1 p' wins over 'p')
5. Sort by order specified
 - latter specified wins

Firebug: CSS debugging

- Firefox add-on (<http://getfirebug.com/>)
- Inspect XHTML/CSS of any page
- Change styles dynamically



Available Formatting

- Font
- Text
- Background
- Display
- Box
- Positioning

Font properties

property	description
font-family	which font will be used
font-size	how large the letters will be drawn
font-style	used to enable/disable italic style
font-weight	used to enable/disable bold style

Font family

```
p {
  font-family: Garamond, "Times New Roman", serif;
}
```

This paragraph uses the above style.

- can specify multiple fonts from highest to lowest priority
- generic font names: serif, sans-serif, cursive, fantasy, monospace

Font weight, style, variant

```
p {
  font-weight: bold;
  font-style: italic;
}
```

This paragraph uses the style above.

- Weight values: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
- Style values: normal | italic | oblique | inherit
- Variant values: normal | small-caps | inherit

Text Properties

property	description
text-align	alignment of text within its element
text-decoration	decorations such as underlining
line-height , word-spacing , letter-spacing	gaps between the various portions of the text
text-indent	indents the first letter of each paragraph

text-align

```
blockquote { text-align: justify; }
h2 { text-align: center; }
```

The Emperor's Quote

[TO LUKE SKYWALKER] The alliance... will die. As will your friends. Good, I can feel your anger. I am unarmed. Take your weapon. Strike me down with all of your hatred and your journey towards the dark side will be complete.

- text-align can be left, right, center, or justify (which widens all full lines of the element so that they occupy its entire width)

text-decoration

```
p { text-decoration: underline; }
```

This paragraph uses the style above.

- can also be overline, line-through, blink, or none
- effects can be combined:


```
text-decoration: overline underline;
```

Background properties

property	description
background-color	color to fill background
background-image	image to place in background
background-position	placement of bg image within element
background-repeat	whether/how bg image should be repeated
background-attachment	whether/bg image scrolls with page
background	shorthand to set all background properties

```
body { background-image: url("images/draft.jpg"); background-repeat: repeat-x; }
```

This is the first paragraph
This is the second paragraph...
It occupies 2 lines

- can be repeat (default), repeat-x, repeat-y, or no-repeat

Display property

property	description
display	sets the type of CSS box model an element is displayed with.

- values: inline | block | list-item | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit

```
<ul id="topmenu">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

```
#topmenu li { display: inline; border: 2px solid gray; margin-right: 1em; }
```

Item 1 Item 2 Item 3

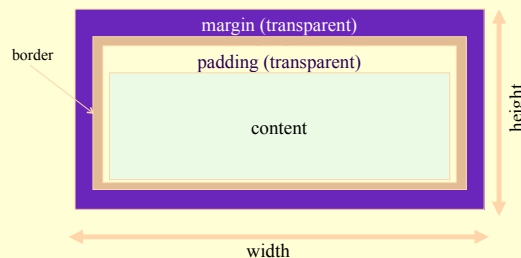
Visibility property

property	description
visibility	sets whether an element should be shown onscreen; can be visible (default) or hidden

```
p.secret { visibility: hidden; }
```

- hidden elements will still take up space onscreen, but will not be shown
 - to make it not take up any space, set display to none instead
- can be used to show/hide dynamic HTML content on the page in response to events

Box Formatting Model



Border properties

```
h2 { border: 5px solid red; }
```

property	description
border	thickness/style/size of border on all 4 sides

- thickness (specified in px, pt, em, or `thin`, `medium`, `thick`)
- style (`none`, `hidden`, `dotted`, `dashed`, `double`, `groove`, `inset`, `outset`, `ridge`, `solid`)
- color (specified as seen previously for text and background colors)

Border example

```
h2 {
  border-left: thick dotted #CC0088;
  border-bottom-color: rgb(0, 128, 128);
  border-bottom-style: double;
}
```

- each side's border properties can be set individually
- if you omit some properties, they receive default values (e.g. border-bottom-width above)

Padding properties

property	description
<code>padding</code>	padding on all 4 sides
<code>padding-bottom</code>	padding on bottom side only
<code>padding-left</code>	padding on left side only
<code>padding-right</code>	padding on right side only
<code>padding-top</code>	padding on top side only

```
p { padding: 20px; border: 3px solid black; }
h2 { padding: 0px; background-color: yellow; }
```

This is the first paragraph

This is the second paragraph

This is a heading

Margin properties

property	description
<code>margin</code>	margin on all 4 sides
<code>margin-bottom</code>	margin on bottom side only
<code>margin-left</code>	margin on left side only
<code>margin-right</code>	margin on right side only
<code>margin-top</code>	margin on top side only

```
p {
  margin: 50px;
  background-color: fuchsia;
}
```

This is the first paragraph

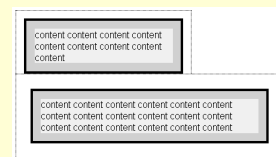
This is the second paragraph

Positioning

- static
- relative
- float
- fixed

static

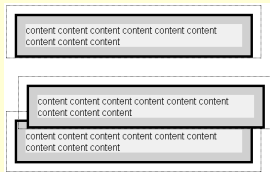
- Default or normal positioning
- Block boxes flow vertically starting at the top of their containing block
- Vertical margins are collapsed



relative

- Initially positioned following the normal flow rules.
- Surrounding boxes are positioned accordingly.
- Box is moved according to its offset properties.
- Offset specified top, right, left and bottom style properties.

```
#div2 {
  position: relative;
  top: 50px;
  left 50px;
}
```

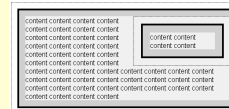


43

float

- Set the float property to left or right.
- Vertical: positioned as with normal flow; top aligned with the top of the current line box.
- Horizontal: shifted to the right or left of containing block.
- Inline content is then allowed to flow around.
- Important:** Need to set width of box

```
<p>
  <span style="float:right,width:40%;">content...</span>
  content content content .....
</p>
```

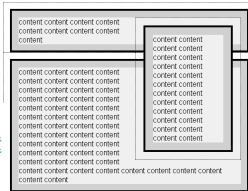


float

- A float can overlaps boxes adjacent to it in the normal flow; does not affect vertical positioning of subsequent boxes.

```
<p>
  <span style="float:right,width:40%;">
    content content content content content
  </span>
  content content content content...
</p>

<p>
  content content content content
  content content content content...
</p>
```



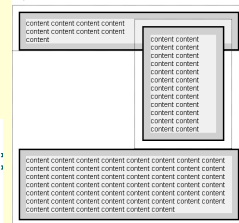
45

clear

- clear property pushes box to the bottom of preceding float box
- Can be set to right, left, both

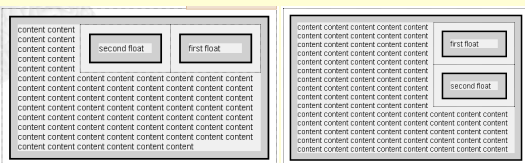
```
<p>
  <span style="float:right,width:40%;">
    content content content content content
  </span>
  content content content content...
</p>

<p style="clear:right;">
  content content content content
  content content content content...
</p>
```



46

In-class Exercise



47

fixed

- Set position relative to browser window
- Determined by offset values: top, right, bottom and left
- A fixed element does not move when a web page is scrolled.
- Have no effect of other boxes in page

48

Best practices

1. if possible, lay out an element by *aligning* its content
 - horizontal alignment: text-align
 - set this on a block element; it aligns the content within it (not the block element itself)
 - vertical alignment: vertical-align
 - set this on an inline element, and it aligns it vertically within its containing element
2. if alignment won't work, try *floating* the element
3. if floating won't work, try *positioning* the element
 - absolute/fixed positioning are a last resort and should not be overused

CSS validation

```
<p>
  <a href="http://jigsaw.w3.org/css-validator/check/referer">
    </a>
</p>
```

- jigsaw.w3.org/css-validator/
- checks your CSS to make sure it meets the official CSS specifications
- more picky than the web browser, which may render malformed CSS correctly