THE UNIVERSITY OF WESTERN AUSTRALIA

Achieving International Excellence

Topic 3 Cascading Style Sheets

CITS3403 Web & Internet Technologies

Reference: Sebesta, Chapter 3

SCHOOL OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

Why Style Sheets?

- Separation of content and presentation ("separation of concerns")
- Not new to the web!
  - eg. WYSIWYG word processors have had considerable problems with consistency
  - stylistic information embedded with content - dependent on user
  - two document styles rarely match
  - not good for example for
    - multiple contributors to a document (eg. promotional booklet)
    - conferences
    - book series publishers
    - etc

Why Style Sheets?

– Compare this with the *LaTeX* Document Typesetting system by Donald Knuth...

```
\documentclass[conference]{IEEEtran}
\usepackage{graphicx,times,psfig,amsmath}

\begin{document}

\title{Direction Matters in High-Dimensional Optimisation}
\author{Cara MacNish, {\em Member, IEEE}, Xin Yao, {\em Fellow,
IEEE}}

\maketitle

\begin{abstract}
Directional biases are evident in many benchmarking problems for real-
valued global optimisation
```

Direction Matters in High-Dimensional Optimisation

Cara MacNish, Member IAENG, Kim Ves Ashlee, IAENG

Abstract— The natural biases or preferences in many benchmark optimisation problems for real-valued global optimisation, as well as many of the evolutionary and allied algorithms that have been proposed for solving them, [illegible] can give misleading views of algorithm performance.

In this paper we study the effects of directional bias for high-dimensional optimisation problems. We show that the impact of directional bias is magnified as dimension increases, and can in some cases lead to differences in performance of many orders of magnitude.

We present a new version of the classical evolutionary programming algorithm, which we call *unbiased evolutionary programming* (UEP), and show that it has markedly improved performance for high-dimensional optimisation

I. Introduction

Many of the problems used to benchmark real-valued global optimisation algorithms are subject to bias [1], which may in turn give misleading results for the algorithms used to solve them. An example that has shown significant performance differences is *directional bias*, most commonly in the form of *axial bias*, where features of a problem surface are aligned with the Cartesian coordinate system in which it is defined. Some solution algorithms are able to exploit this alignment. It has been shown in [2], [3], for example, that traditional binary encoded genetic algorithms show improved performance on functions with axial bias, and that performance simultaneous with the function is rotated with respect to the co-ordinate system.

An extreme example of directional bias a problem that are *linearly separable*. An n-dimensional function $f : R^n \mapsto R$ is linearly separable iff:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} f_i(x_i)$$

for some f_i (which may be non-linear) that is it can be defined as a sum of functions of the individual variables. More generally, a function can be defined as decomposable or separable [4] iff:

$$\arg\min_{x} f(x_1, \ldots, x_n) = (\arg\min_{x_1} f(x_1, \ldots), \ldots, \arg\min_{x_n} f(\ldots, x_n))$$

In this paper we study the effects of directional bias for *high-dimensional* optimisation problems by examining variants of a classical evolutionary programming algorithm [5], [6], [7]. Following [8] we are particularly interested in the range 100 to 1000 dimensions, although many of the trends in our results continue to higher dimensions.

We study two primary variants of the classical algorithm. The first is designed to exploit separability, and demonstrates that such high-dimensional problems can be beneficially reduced to a number of 1-dimensional problems. This highlights the synergy gains that a more "generalist" algorithm needs to make to outperform this reductionist approach.

The second is a new algorithm designed to evolve without directional bias. We show that this leads to significant performance improvements over the classical algorithm, in some cases several orders of magnitude within the range of dimensions studied.

In order to further elucidate the specific difference in performance between the algorithms, we additionally examine three "intermediate" algorithms that share features of the primary algorithms. We find, however, that the unbiased algorithm continues to significantly outperform the others on non-separable problems, and all but the reductionist algorithm on separable problems, at high dimensions.

The algorithms studied in this paper are described in Section II. The problem set used to study the algorithms is taken from [8], and is briefly described in Section III. Section IV summarises the main results of the study and discusses their implications. The paper is concluded in Section V.

II. Algorithms

In this paper we are concerned with bounded, constrained, real-valued optimisation. A problem is a pair (S, f) where $S \subset R^n$ is a bounded set on R^n, and $f : S \mapsto R$ is an n-dimensional *fitness function*. Without loss of generality we assume all problems are cast as minimisation problems. Briefly our goal is to find a point $x_{min} \in S$ such that $f(x_{min})$ is a global minimum on S, that is

$$\forall x \in S \cdot f(x_{min}) \leq f(x).$$

In practice, the goal of our algorithm will be to find the smallest value achieved by x_{min} that is one before reaching a stopping criterion.

Why Style Sheets?

- Style set globally
  - All conference papers in a proceedings, chapter in a book, books in a series, etc have the same style.
- More flexible, more control
- Later decide to change the style, just change stylesheet and *all documents change accordingly*
- Similar ideas to software engineering principles of modularity and encapsulation
  - one person can work on the layout and style
  - different people can contribute content
  - style re-use or adaptation
- Next: CSS...

What is CSS?

- Not to be confused with the Brazilian rock band of the same name...

http://flickr.com/photos/94038853@N00/1240577104

CSS (Cansei de Ser Sexy) performing at Rock en Seine in August 2007

What is CSS?

- CSS stands for *Cascading Style Sheets*
  - stylesheet language for web
  - used to specify the presentation (layout and style) of markup languages
  - can be applied to any XML document (including XHTML)
  - superceded many HTML attributes that mixed presentation with content

Why CSS?

- Separation of content and presentation
- Advantages for the web
  - *Speed* - stylesheet(s) downloaded once, rather than with each page (if content and style information is intermingled)
  - *Maintainability* - can be "centrally" maintained, easier to update
  - *Accessibility* - can make pages appear similar on different browsers and devices
  - *Portability* - eg. printing, porting to new devices
  - *Reduced work* - eg. don't have to specify alignment every time an element is used
  - *Consistency* - make an organisation's web pages have consistent "look and feel" - corporate ID, brand (and update as brand updates)
    - eg. UWA...

THE UNIVERSITY OF WESTERN AUSTRALIA

Faculty of Natural & Agricultural Sciences

Faculty Home | Faculty Intranet | Ag & Resource Econs | Animal Biology | Earth & Geographical | Plant Biology

Search UWA Website for keywords... Go

Information For
- Prospective Undergradua...
- Prospective Postgraduates
- Current Students
- Alumni
- Staff

Information About
- The Faculty
- Faculty Research

THE FACULTY OF Natural & Agricultural Sciences

Welcome

Science was a founding faculty of the University of

NEWS

UWA EXPO 2008
READY TO HIT TOP GEAR?

See our displays and

THE UNIVERSITY OF WESTERN AUSTRALIA

FACULTY OF ENGINEERING, COMPUTING AND MATHEMATICS

Quick Links

Faculty Home | StudentNet | StaffNet

Site Search UWA Website GO

UWA Home > Faculty of Engineering, Computing and Mathematics > Home

- Courses
- Research
- Business and industry
- Alumni
- StudentNet
- StaffNet
- Contact us

Engineering, Computing and Mathematics

Our Faculty offers exciting and challenging opportunities and a secure future to women and men ready to take on the challenges of the millennium.

Don't just go out into the world. Improve it.

History

- Stylesheet specification administered by W3C (World Wide Web Consortium): http://www.w3.org/Style/CSS/
- CSS releases - see http://www.w3.org/Style/CSS/#specs
  - CSS1 released 1996 - fonts, margins, colours, etc
  - CSS2 released 1998 - absolute positioning, etc
  - CSS3 is on its way - vertical text, user interaction, speech, etc
  - CSS Mobile profile
  - CSS Print profile
  - CSS TV profile

Cascading Style Sheets

- **CSS1** introduced styles for the following document features:
  - Fonts
  - Text
  - Color
  - Backgrounds
  - Block-level Elements

Cascading Style Sheets

- **CSS2** introduced styles for the following document features:
  - Positioning
  - Visual Formatting
  - Media Types
  - Interfaces

Cascading Style Sheets

- **CSS3** (which is still in development) will introduce styles for the following document features:
  - User Interfaces
  - Accessibility
  - Columnar layout
  - International Features
  - Mobile Devices
  - Scalable Vector Graphics

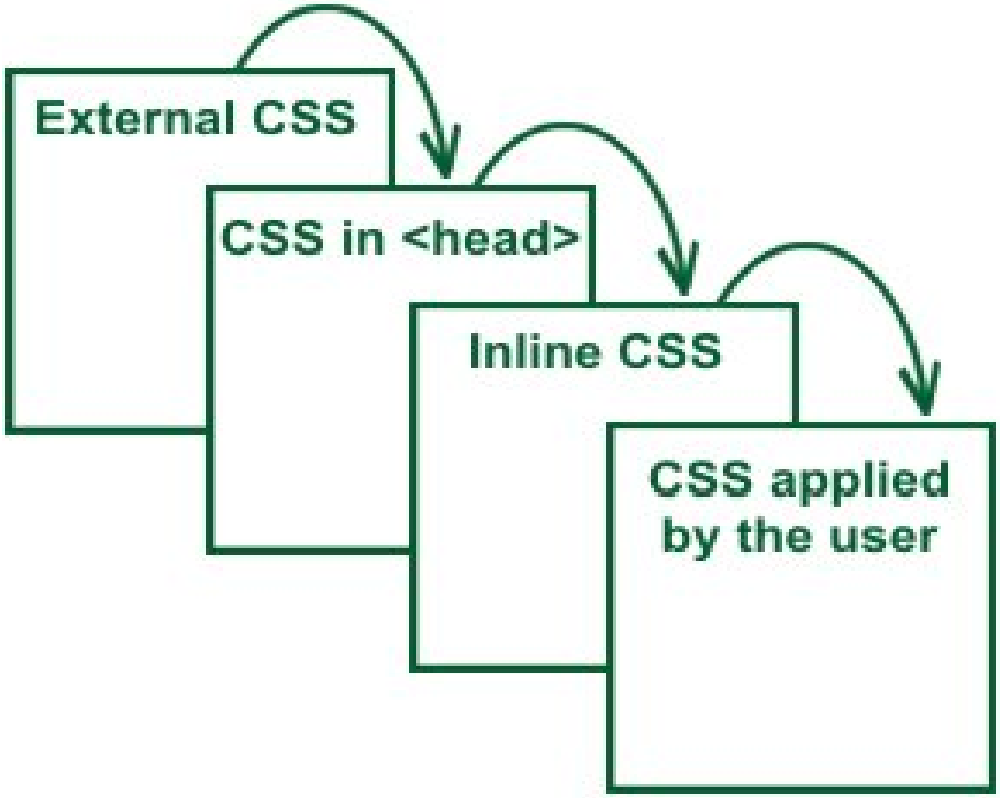
CSS3

- **Borders**
  - border-color
  - border-image
  - border-radius
  - box-shadow
- **Color**
  - HSL colors
  - HSLA colors
  - opacity
  - RGBA colors
- **Text effects**
  - text-shadow
  - text-overflow
  - word-wrap
- **User-interface**
  - box-sizing
  - resize
  - outline
  - nav-top, nav-right, nav-bottom, nav-left
- **Selectors**
  - attribute selectors
- **Basic box model**
  - overflow-x, overflow-y
- **Generated Content**
  - content
- **Other modules**
  - media queries
  - multi-column layout
  - Web fonts

Why "Cascading"?

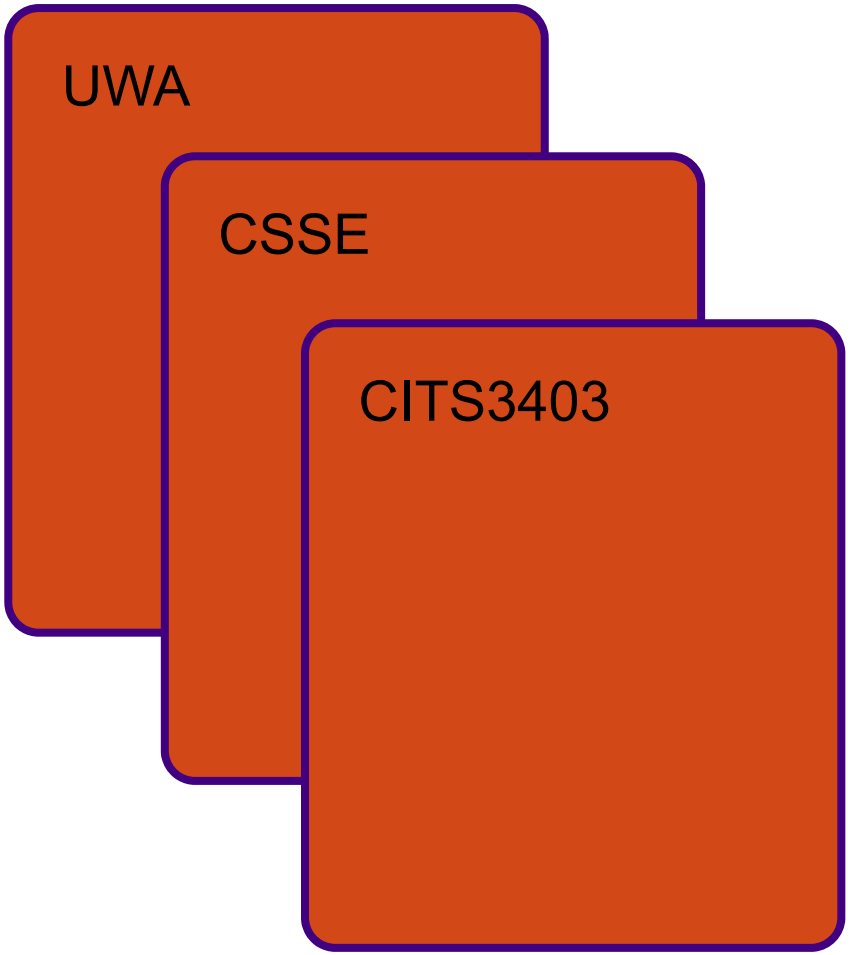
- There are three levels of style sheets
  - *Inline* - specified for a specific occurrence of a tag and apply only to that tag
    – This is fine-grain style, which defeats the purpose of style sheets - uniform style
  - *Document-level* style sheets - apply to the whole document in which they appear
  - *External* style sheets - can be applied to any number of documents
- When more than one style sheet applies to a specific tag in a document, the lowest level style sheet has precedence
  - In a sense, the browser searches for a style property spec, starting with inline, until it finds one (or there isn't one)

Why "Cascading"?



Picture: http://paulbohman.com/web/css/why_cascading

Why "Cascading"?



Linking an External Stylesheet

- A `<link>` tag is used to specify that the browser is to fetch and use an external style sheet file

```
<link rel="stylesheet"  type="text/css"
   href="http://www.wherever.org/termpaper.css">
</link>
```

  - eg. Wikipedia style sheet

http://en.wikipedia.org/wiki/Cansei_de_Ser_Sexy

http://en.wikipedia.org//skins-1.5/common/shared.css?165

- External style sheets can be validated

http://jigsaw.w3.org/css-validator/

Levels of Style Sheets (continued)

- Inline style sheets appear in the tag itself
- Document-level style sheets appear in the head of the document
- External style sheets are in separate files, potentially on any server on the Internet
  - Written as text files with the MIME type `text/css`

In-line Style Specification Format

- Style specification appears as the value of the `style` *attribute*
  - General form:

```
style = "property_1: value_1; property_2: value_2;…
         property_n: value_n"
```

  - Example:

```
<p style="background: purple; color: white;">
  This paragraph will have white text on a purple
  background.
</p>
```

Document-level Format

- Style specification appears as a list of rules that are the *content* of a `<style>` tag
- The `<style>` tag must include the `type` attribute, set to `"text/css"`
- Contained in the document <head>

Document-level Format

- General form:

```
<style type = "text/css">
 <!--
     rule list
 -->
</style>
```

- Form of the rules:

  selector {list of property/values}

  – Each property/value pair has the form:
    property: value

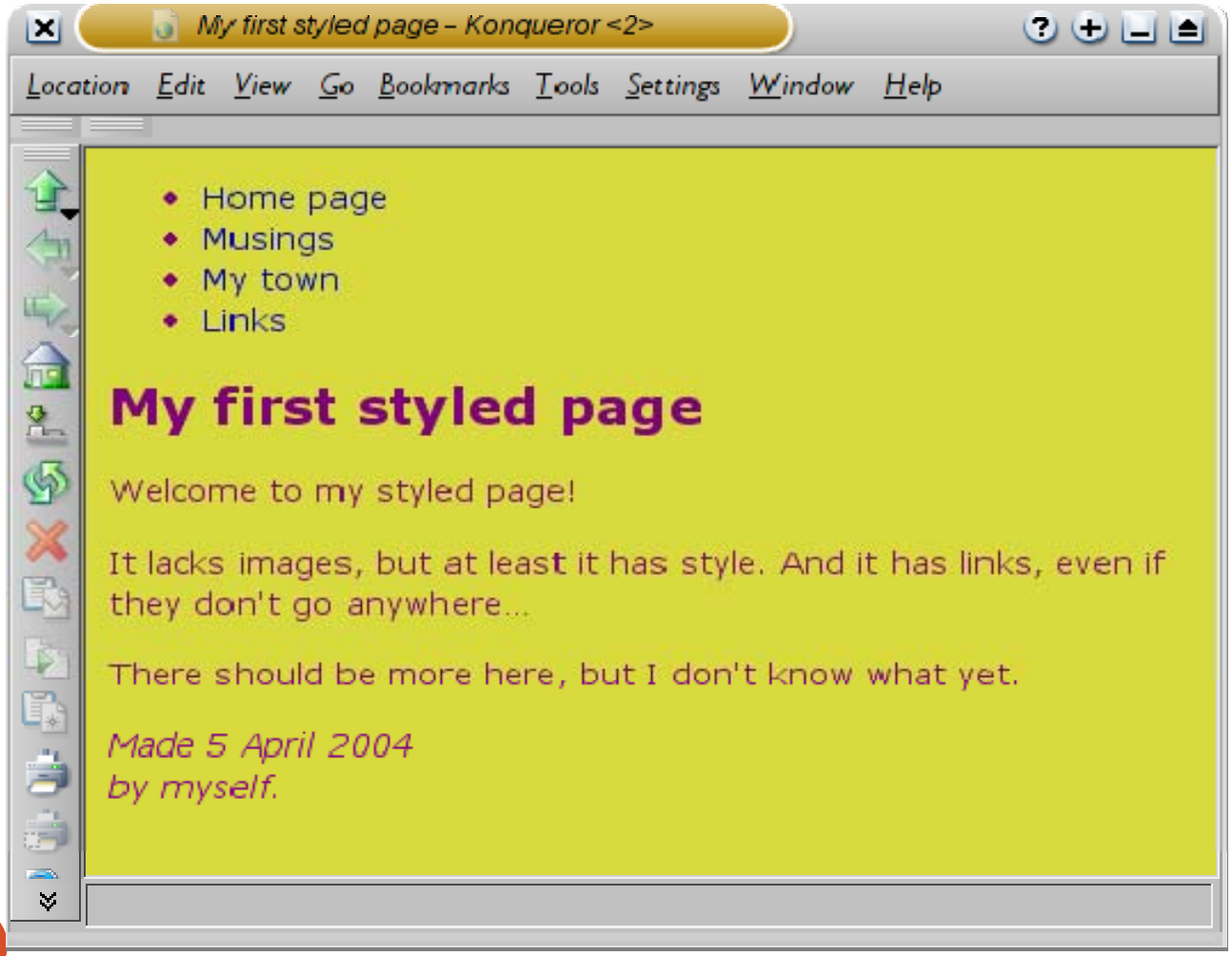
  – Pairs are separated by semicolons, just as in the value of a `<style>` tag

Document-level Format

- Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>My first styled page</title>
  <style type="text/css">
      body {
            color: purple;
            background-color: #d8da3d
      }
  </style>
</head>

<body>

[etc.]
```

Document-level Format



External style sheet format

- Form is a list of style rules

  selector {list of property/values}

  as in the content of a <style> tag for document-level style sheets

```
body {
  color: purple;
  background-color: #d8da3d
}
```

Selecting Elements

- There are numerous ways of specifying to which elements style rules apply. Here are examples of some of the more commonly used:

p {color:red}	Every p element
h1,h2,h3 {…}	Group selector
strong em {…}	Contextual selector
div[secret="yes"] {…}	Attribute selector
span.important {…}	Class selector
p#1234 {…}	ID selector

Selector Forms: Simple

- The *selector* is a tag name or a list of tag names, separated by commas, eg:

```
h1, h2 {font-size: 24pt}
```

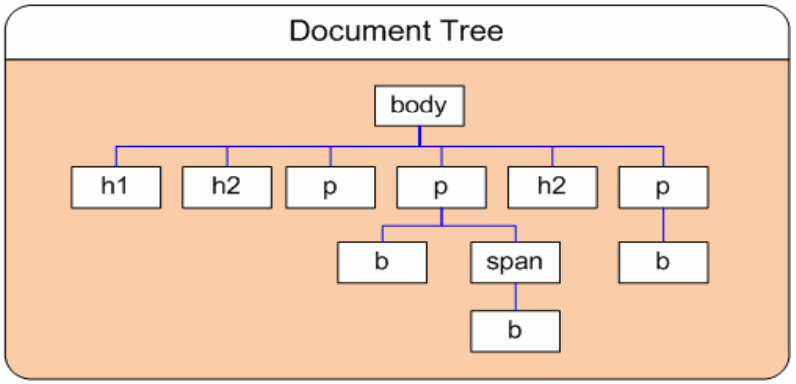
- *Contextual (or descendant) selectors, eg:*

```
body b em {font-size: 14pt}
```

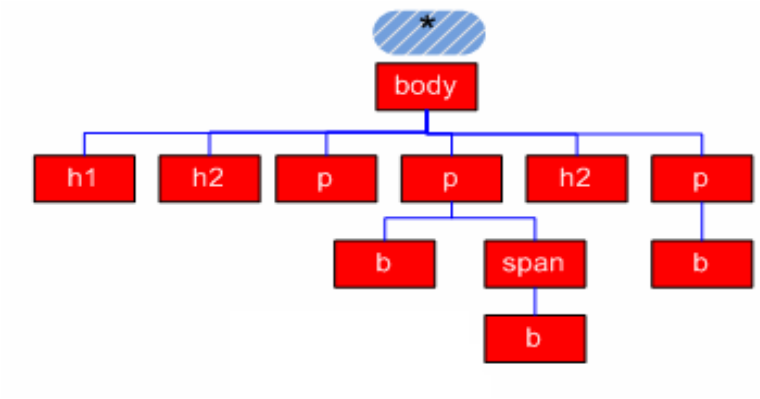
Selector	Matches
*	Any element in the hierarchy
e	The specified element in the hierarchy, where *e* is the specified element
e1, e2, e3, ...	The group of elements *e1, e2, e3, ...*
e f	The element *f* when it is a descendant of the element *e*
e > f	The element *f* when it is a direct child of the element *e*
e + f	The element *f* when it is immediately preceded by the sibling element *e*

Selector: Any element

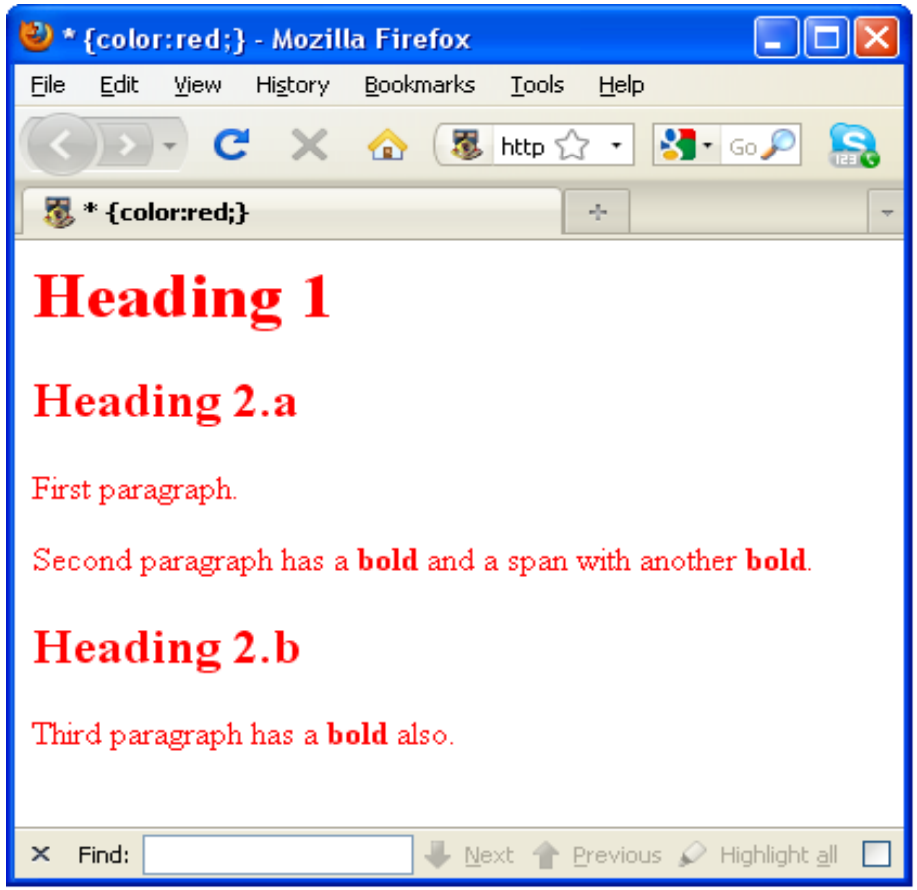
```
<html>
    <head>
        <style type="text/css">
        <!–
            * {color:red;}
        -->
        </style>
    </head>
    <body>
        <h1> Heading 1</h1>
        <h2> Heading 2.a</h2>
        <p> First paragraph. </p>
        <p> Second paragraph has a <b>bold</b>  and a <span>span with another <b>bold</b></span>.
    </p>
        <h2> Heading 2.b</h2>
        <p> Third paragraph has a <b>bold</b> also. </p>
```



Selector: Any element

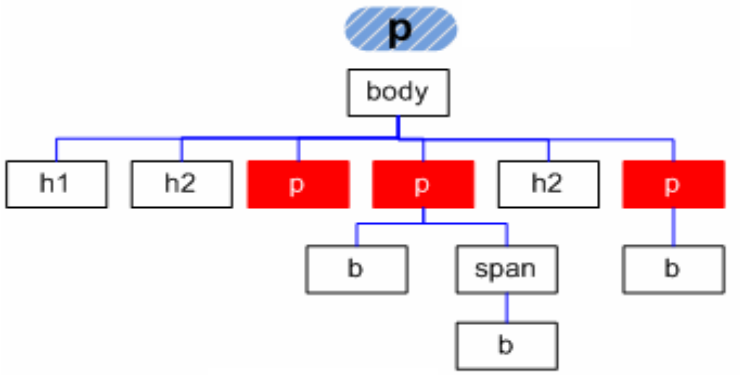


Any selector example

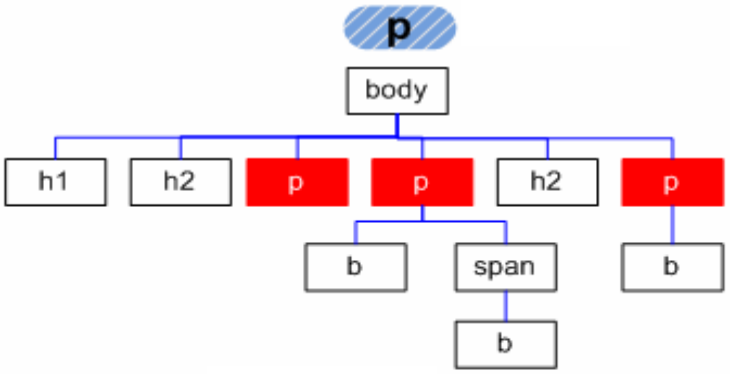


Example: Selector p

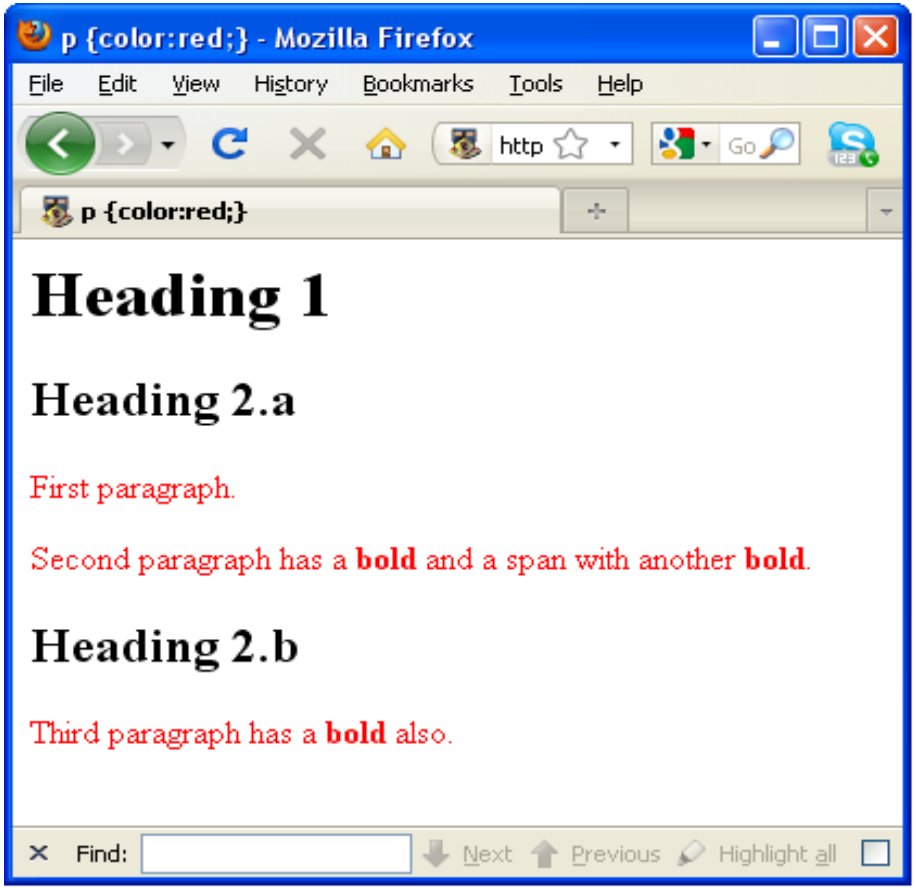
```
<html>
    <head>
        <style type="text/css">
        <!–
          p {color:red;}
        -->
        </style>
    </head>
    <body>
        <h1> Heading 1</h1>
        <h2> Heading 2.a</h2>
        <p> First paragraph. </p>
        <p> Second paragraph has a <b>bold</b>  and a <span>span with another<b>bold</b></span>. </p>
        <h2> Heading 2.b</h2>
        <p> Third paragraph has a <b>bold</b> also. </p>
    </body>
```



Example: Selector p

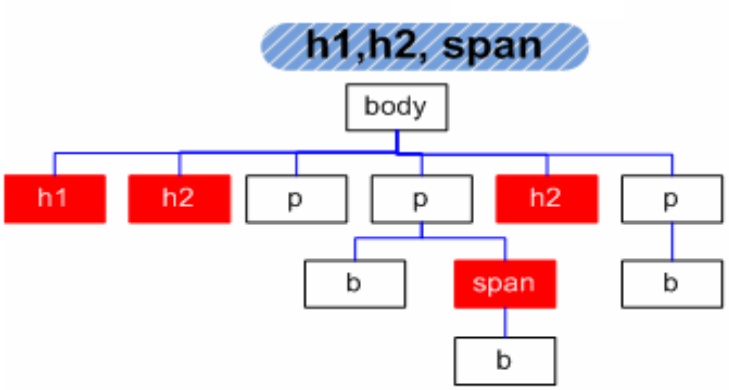


Selector p example

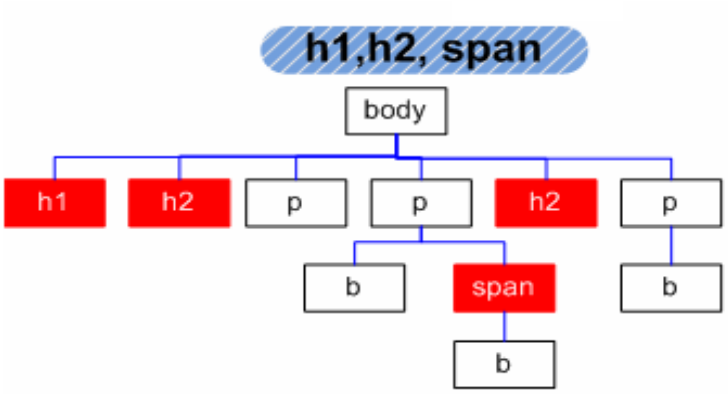


Example: Selector h1,h2,span

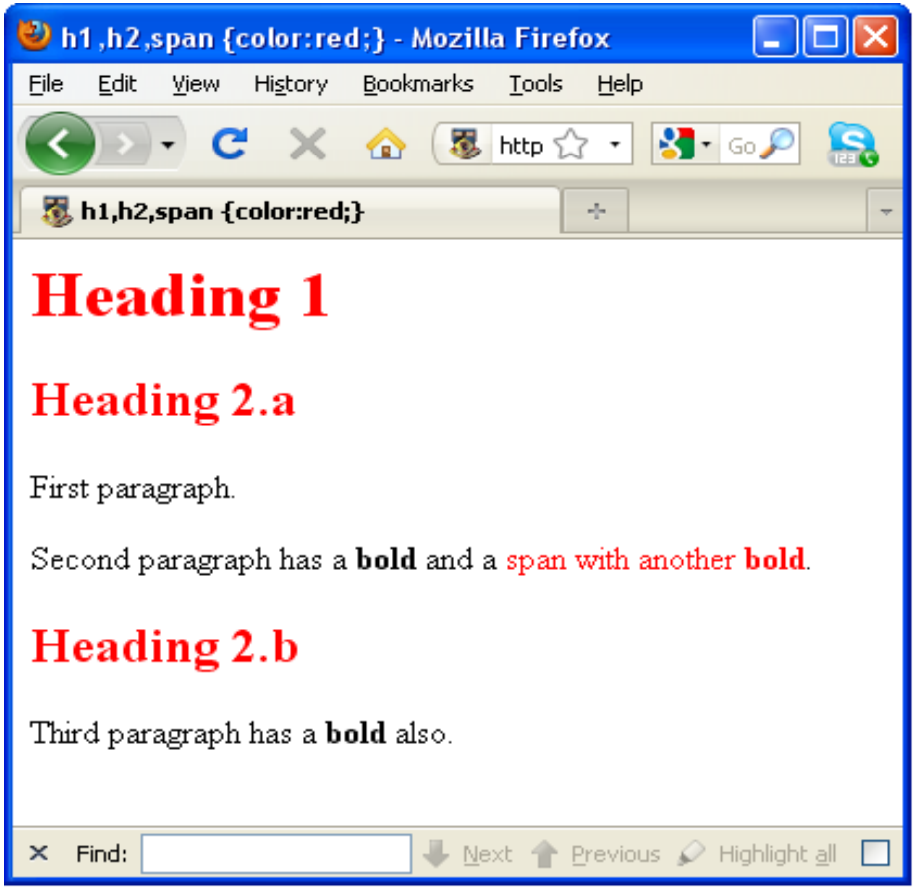
```
<html>
    <head>
        <style type="text/css">
        <!–
            h1,h2,span {color:red;}
        -->
        </style>
    </head>
    <body>
        <h1> Heading 1</h1>
        <h2> Heading 2.a</h2>
        <p> First paragraph. </p>
        <p> Second paragraph has a <b>bold</b>  and a <span>span with another <b>bold</b></span>.
    </p>
        <h2> Heading 2.b</h2>
        <p> Third paragraph has a <b>bold</b> also. </p>
```



Example: Selector h1,h2,span

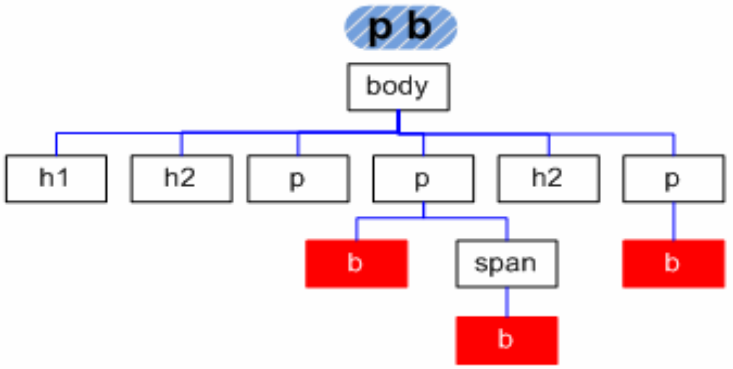


Selector h1,h2,span example

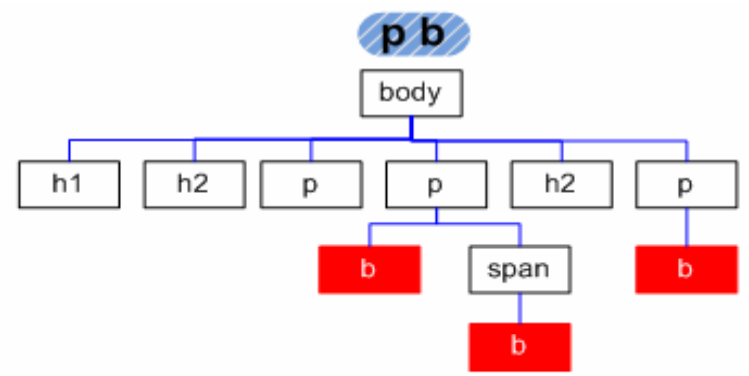


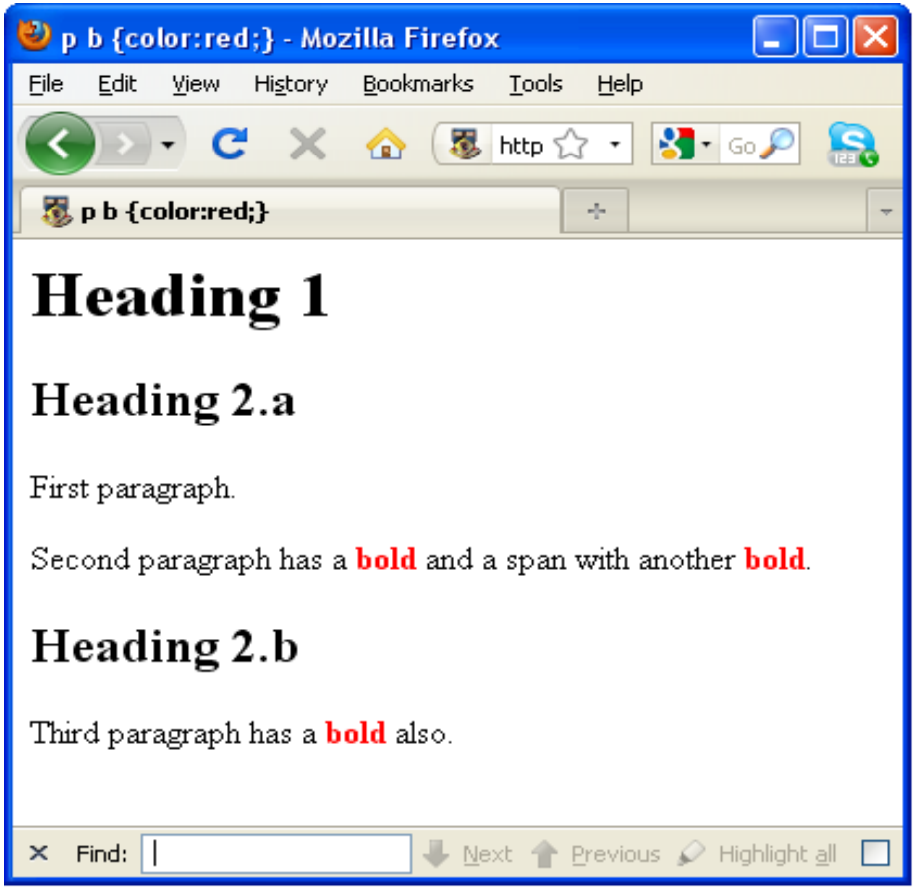
Example: Selector p b

```
<html>
    <head>
            <style type="text/css">
            <!–
                p b {color:red;}
            -->
            </style>
    </head>
    <body>
            <h1> Heading 1</h1>
            <h2> Heading 2.a</h2>
            <p> First paragraph. </p>
            <p> Second paragraph has a <b>bold</b>  and a <span>span with another <b>bold</b></span>.
    </p>
            <h2> Heading 2.b</h2>
            <p> Third paragraph has a <b>bold</b> also. </p>
```



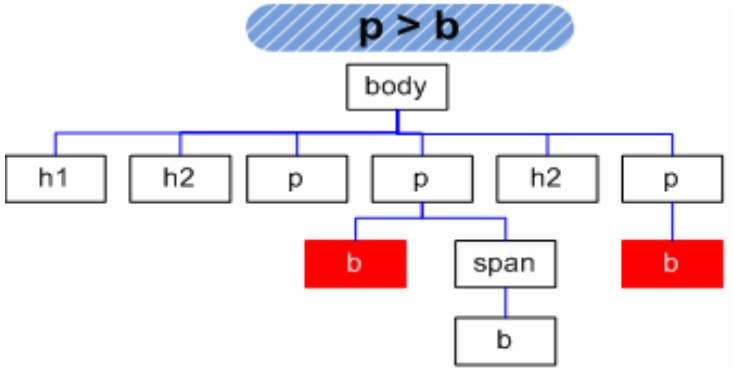
Example: Selector p b



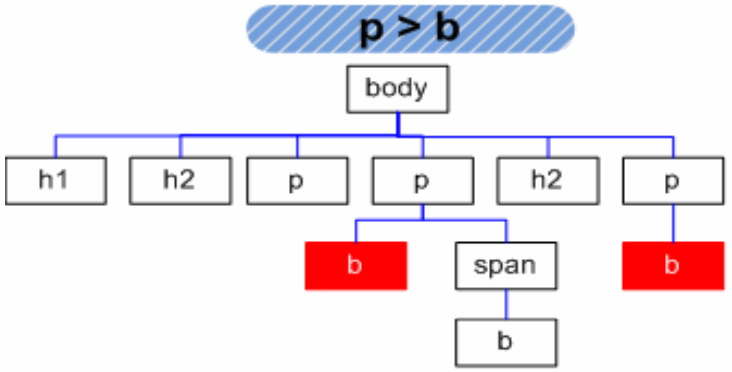


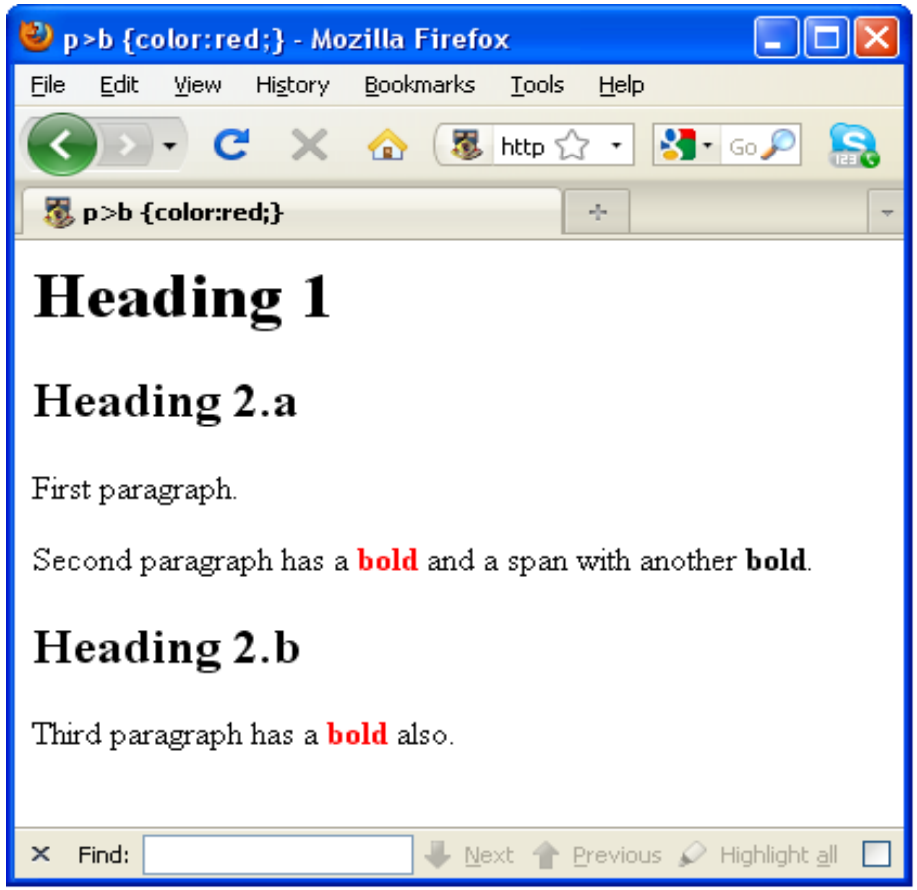
Example: Selector p>b

```
<html>
    <head>
        <style type="text/css">
        <!–
            p>b {color:red;}
        -->
        </style>
    </head>
    <body>
        <h1> Heading 1</h1>
        <h2> Heading 2.a</h2>
        <p> First paragraph. </p>
        <p> Second paragraph has a <b>bold</b>
                and a <span>span with another <b>bold</b></span>. </p>
        <h2> Heading 2.b</h2>
        <p> Third paragraph has a <b>bold</b> also. </p>
```



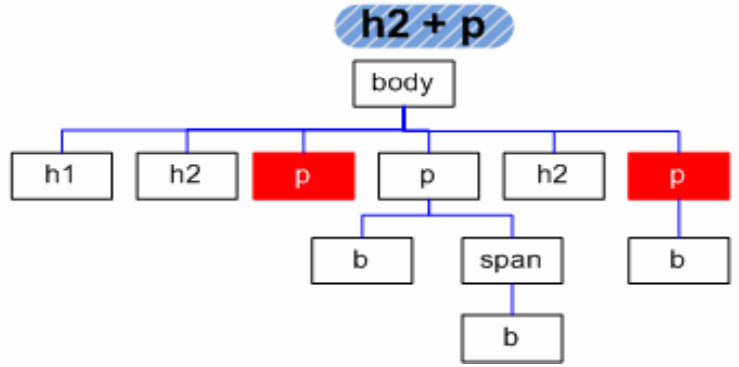
Example: Selector p>b



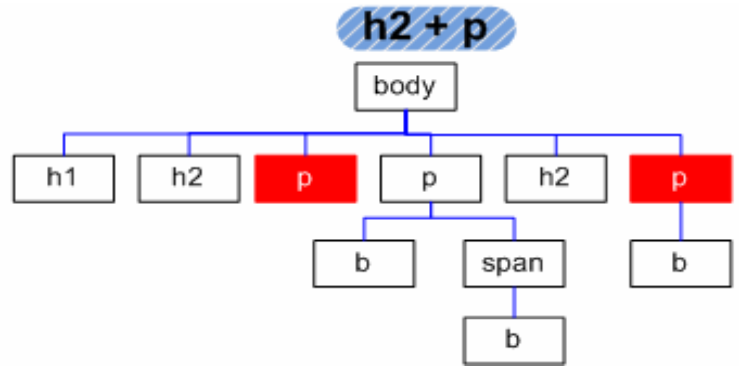


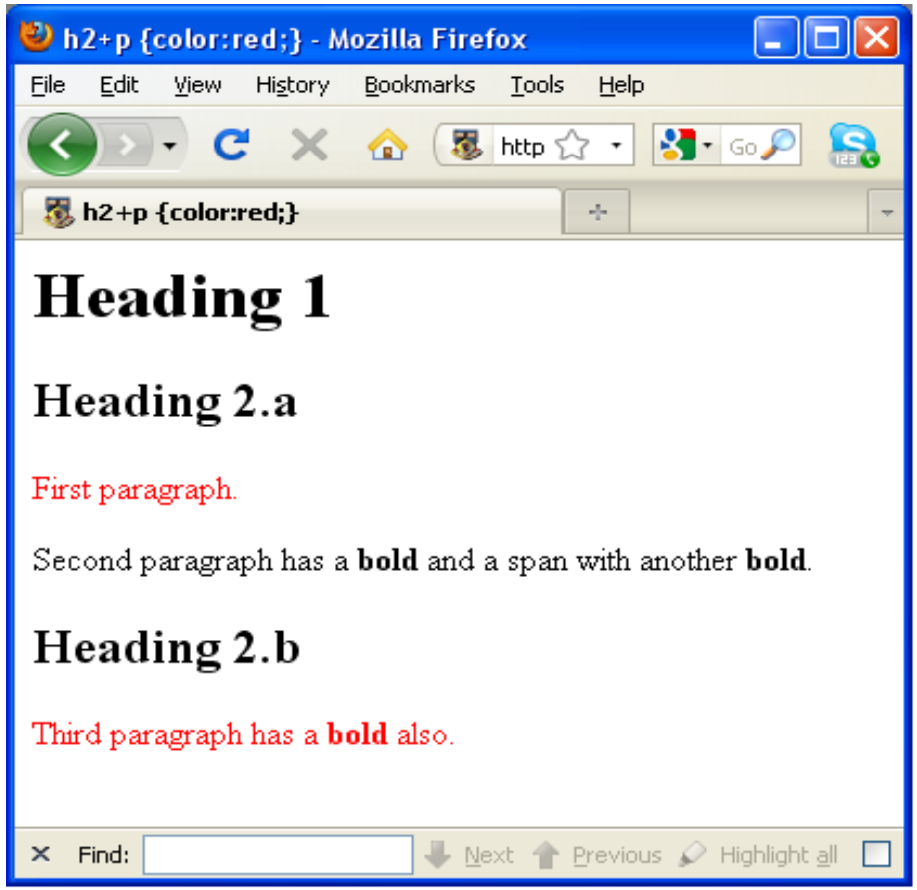
Example: Selector h2+p

```
<html>
    <head>
        <style type="text/css">
        <!–
            h2+p {color:red;}
        -->
        </style>
    </head>
    <body>
        <h1> Heading 1</h1>
        <h2> Heading 2.a</h2>
        <p> First paragraph. </p>
        <p> Second paragraph has a <b>bold</b>  and a <span>span with another <b>bold</b></span>.
    </p>
        <h2> Heading 2.b</h2>
        <p> Third paragraph has a <b>bold</b> also. </p>
```

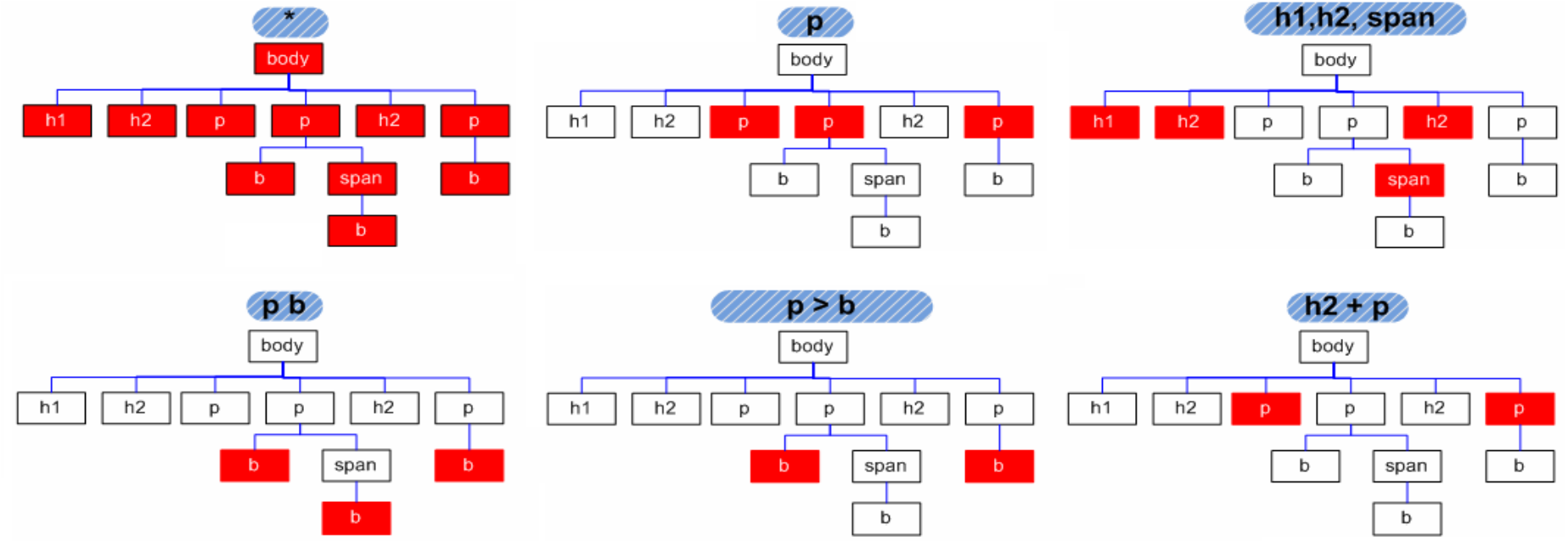


Example: Selector h2+p





Examples Summary



Class Selectors

- Used to allow different occurrences of the same tag to use different style specifications
- A style *class* has a name, which is attached to a tag name

```
p.narrow {property/value list}
p.wide {property/value list}
```

Class Selectors

- The class you want on a particular occurrence of a tag is specified with the class attribute of the tag
- For example:

```
<p class = "narrow">
   ...
</p>
...

<p class = "wide">
   ...
</p>
```

Generic Selectors

- A *generic class* can be defined if you want a style to apply to more than one kind of tag
- A generic class name must begin with a period
- Example,

```
.really-big {font-size: 60pt; ...}
```

- Use it as if it were a normal style class

```
<h1 class = "really-big"> … </h1>
...
<p class = "really-big"> … </p>
```

id Selectors

- An *id* selector allows the application of a style to one specific element
- General form:

```
#specific-id {property/value list}
```

- Example:

```
#breadcrumbs {
  top: 60px; width: 100%; height: 23px;
  text-indent: 15px; padding-top: 1px;
  color: white;
}
```

id Selectors

```
<p id="breadcrumbs">

   <a href="http://web.csse.uwa.edu.au/">School Home</a> |
     <a href="http://web.csse.uwa.edu.au/current/">Current
       Students</a> |
     <a href="http://undergraduate.csse.uwa.edu.au/units/
       CITS4230/">Internet Technologies</a>

</p>
```

Pseudo Classes

- Pseudo classes are styles that apply when something happens, rather than because the target element simply exists
- Names begin with colons
  - `:hover` classes apply when the mouse cursor is over the element
  - `:focus` classes apply when an element has focus

Pseudo Class Example

```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
     <title> Checkboxes </title>
    <style type = "text/css">
      input:hover {color: red;}
      input:focus {color: green;}
    </style>
  </head>
  <body>
    <form action = "">
      <p>  Your name:   <input type = "text" />    </p>
    </form>
  </body>
</html>
```

Conflict Resolution

- When two or more rules apply to the same tag there are rules for deciding which rule applies
- Document level
  - In-line style sheets have precedence over document style sheets
  - Document style sheets have precedence over external style sheets
- Within the same level there can be conflicts
  - A tag may be used twice as a selector
  - A tag may inherit a property and also be used as a selector
- Style sheets can have different sources
  - The author of a document may specify styles
  - The user, through browser settings, may specify styles
- Individual properties can be specified as important

Precedence Rules

- From highest to lowest

1. Important declarations with user origin
2. Important declarations with author origin
3. Normal declarations with author origin
4. Normal declarations with user origin
5. Any declarations with browser (or other user agent) origin

Tie-Breakers

- Specificity

  1. id selectors
  2. Class and pseudo-class selectors
  3. Contextual selectors
  4. General selectors

- Position
  - Essentially, later has precedence over earlier

Properties

- CSS1 includes 60 different properties in 7 categories:
  - Fonts
  - Lists
  - Alignment of text
  - Margins
  - Colors
  - Backgrounds
  - Borders

Property Values

- Keywords - left, small, …
  - Not case sensitive
- Length - numbers, possibly with decimal points
- Units:
  - px - pixels
  - in - inches, cm - centimeters, mm - millimeters
  - pt - points
  - pc - picas (12 points)
  - em - height of the letter ‘m’
  - ex-height - height of the letter ‘x’
  - No space is allowed between the number and the unit specification
    e.g., `1.5 in` is illegal

Property Values (continued)

- Percentage - just a number followed immediately by a percent sign
- URL values
  - url(protocol://server/pathname)
- Colors
  - Color name
  - rgb(n1, n2, n3)
    - Numbers can be decimal or percentages
  - Hex form: #XXXXXX
- Property values are inherited by all nested tags, unless overridden

Font Properties

Generic Name	Examples
serif	Times New Roman, Garamond
sans-serif	MS Arial, Helvetica
cursive	Caflisch Script, Zapf-Chancery
fantasy	Critter, Cottonwood
monospace	Courier, Prestige

Font Properties (continued)

- `font-size`
  - Possible values: a length number or a name, such as `smaller`, `xx-large`, etc.
- `font-style`
  - `italic`, `oblique` (useless), `normal`
- `font-weight` - degrees of boldness
  - `bolder`, `lighter`, `bold`, `normal`
    - Could specify as a multiple of 100 (100 – 900)
- `font`
  - For specifying a list of font properties
  - `font: bolder 14pt Arial Helvetica`
  - Order must be: style, weight, size, name(s)
- Examples: fonts.html, fonts2.html

3.6 Font Properties (continued)

- The `text-decoration` property
  - `line-through`, `overline`, `underline`, `none`
  - `letter-spacing` – value is any length property value

List properties

- `list-style-type`
- *Unordered lists*
  - Bullet can be a disc (default), a square, or a circle
  - Set it on either the `<ul>` or `<li>` tag
    - On `<ul>`, it applies to list items..

```
<h3> Some Common Single-Engine Ai
  <ul style = "list-style-type: s
      <li> Cessna Skyhawk </li>
      <li> Beechcraft Bonanza </l
      <li> Piper Cherokee </li>
  </ul>
```

Some Common Single-Engine Aircraft

- Cessna Skyhawk
- Beechcraft Bonanza
- Piper Cherokee

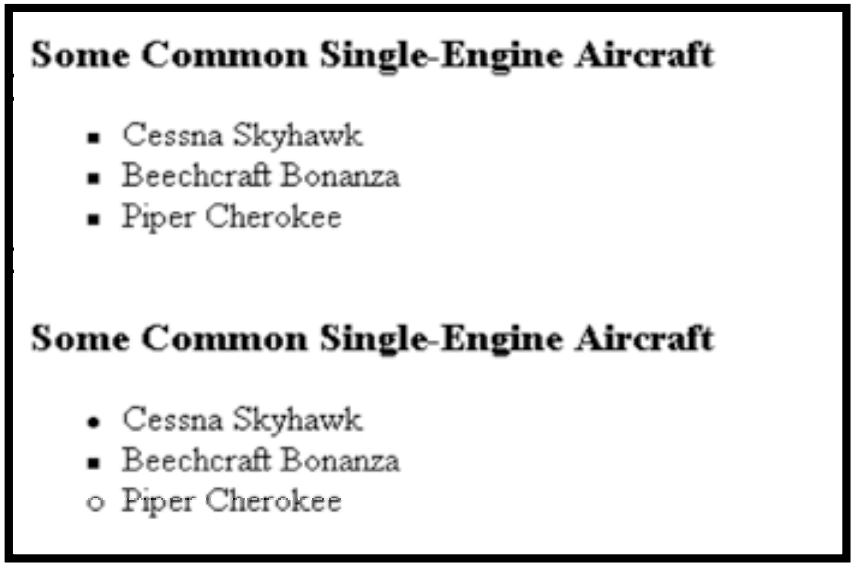
Some Common Single-Engine Aircraft

- Cessna Skyhawk
- Beechcraft Bonanza
- Piper Cherokee

List properties

- On <li>, list-style-type applies to just that item

```
<h3> Some Common Single-Engine Aircraft </h3>
 <ul>
     <li style = "list-style-type: disc">
          Cessna Skyhawk </li>
     <li style = "list-style-type: s
          Beechcraft Bonanza </li>
     <li style = "list-style-type: c
          Piper Cherokee </li>
 </ul>
```



List properties (continued)

- Can use an image for the bullets in an unordered list
  - Example:

```
<li style = "list-style-image: url(bird.jpg)">
```

- On *ordered lists* - `list-style-type` can be used to change the sequence values

Property Values	Sequence Type	First Four Values
`decimal`	Arabic numerals	1, 2, 3, 4
`upper-alpha`	Uppercase letters	A, B, C, D
`lower-alpha`	Lowercase letters	a, b, c, d
`upper-roman`	Uppercase Roman numerals	I, II, III, IV
`lower-roman`	Lowercase Roman numerals	i, ii, iii, iv

→ Example: `sequence_types.html`

- CSS2 has more, like `lower-greek` and `hebrew`

Colors

- Color is a problem for the Web for two reasons:
  1. Monitors vary widely
  2. Browsers vary widely
- There are three color collections
  1. There is a set of 16 colors that are guaranteed to be displayable by all graphical browsers on all color monitors

Name	Hexadecimal Code	Name	Hexadecimal Code
black	000000	green	008000
silver	C0C0C0	lime	00FF00
gray	808080	olive	808000
white	FFFFFF	yellow	FFFF00
maroon	800000	navy	000080
red	FF0000	blue	0000FF
purple	800080	teal	008080
fuchsia	FF00FF	aqua	00FFFF

Colors

2. There is a much larger set, the *Web Palette*
   - 216 *named* colors

     http://www.w3schools.com/html/html_colornames.asp

   - Also in Appendix B of Sebesta text.

3. Any one of 16 million different colors
   - #000000, #000001, #000002, . . . , #FFFFFE, #FFFFFF

Colors

- The `color` property specifies the foreground colour of elements

```
<style type = “text/css”>
  th.red {color: red}
  th.orange {color: orange}
</style>
…
<table border = "5">
  <tr>
    <th class = "red"> Apple </th>
    <th class = "orange"> Orange </th>
    <th class = "orange"> Screwdriver </th>
  </tr>
</table>
```

- The `background-color` property specifies the background color of elements

→ Example: back_color.html

Alignment of Text

- The `text-indent` property allows indentation
  - Takes either a length or a % value
- The `text-align` property has the possible values, `left` (the default), `center`, `right`, or `justify`
- Sometimes we want text to flow around another element - the `float` property
  - The `float` property has the possible values, `left`, `right`, and `none` (the default)
  - If we have an element we want on the right, with text flowing on its left, we use the default `text-align` value (`left`) for the text and the `right` value for `float` on the element we want on the right

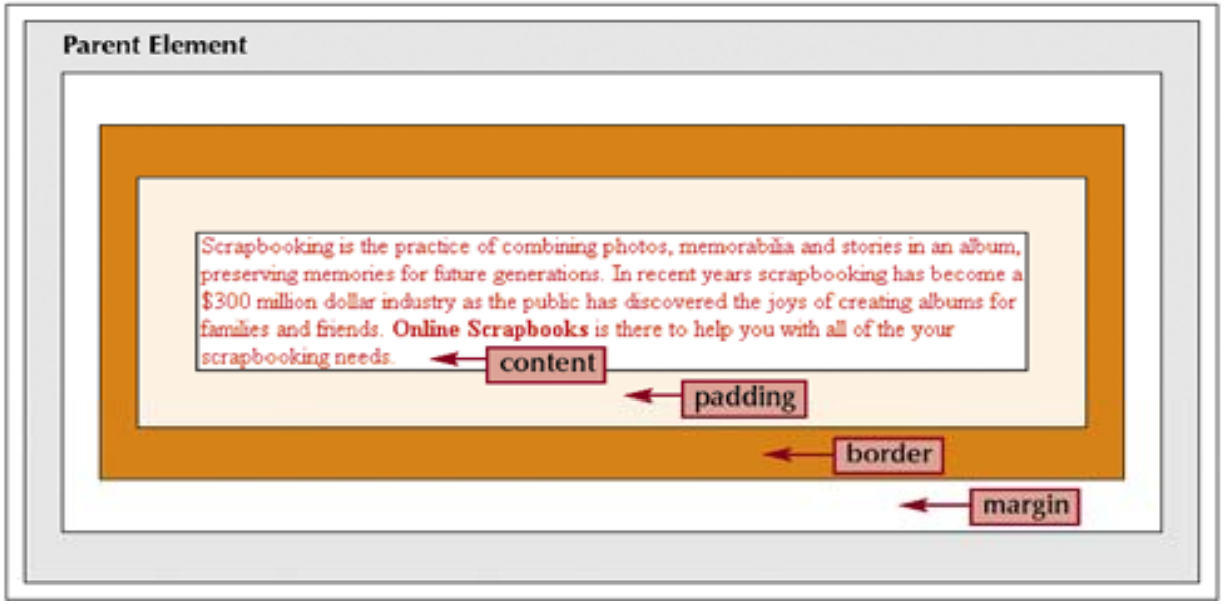
Alignment of Text

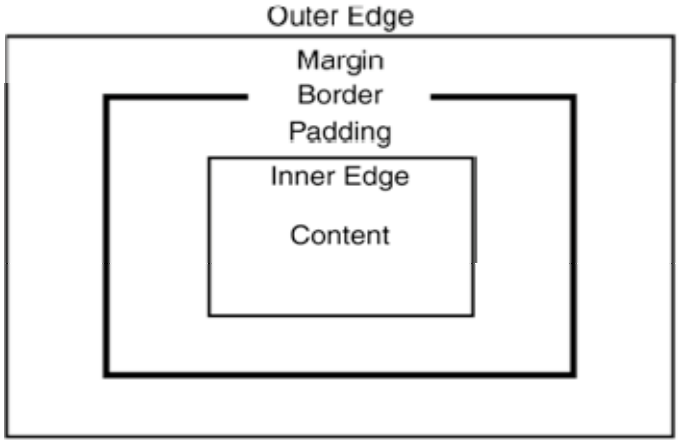
```
<img src = "c210.jpg"
          style = "float: right" />
```

- Some text with the default alignment - `left`

Working with the Box Model

- The **box model** is an element composed of four sections:
  - Margin
  - Border
  - Padding
  - content





The Box Model

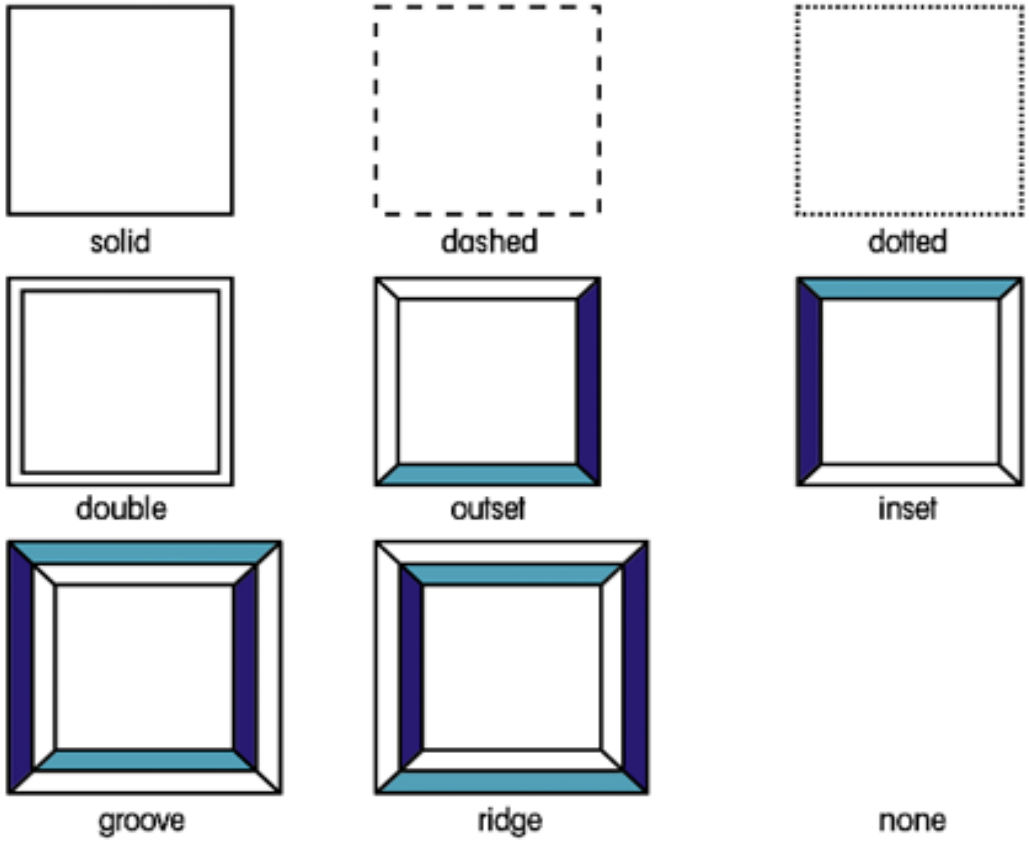
- Borders – every element has a `border-style` property
  - Controls whether the element has a border and if so, the style of the border
  - `border-style` values: `none`, `dotted`, `dashed`, and `double`
  - `border-width: thin`, `medium` (default), `thick`, or a length value in pixels
  - Border width can be specified for any of the four borders (e.g., `border-top-width`)
  - `border-color`: any color
  - Border color can be specified for any of the four borders (e.g., `border-top-color`)

ᘓ → Example: `borders.html`

Border Styles

Border Style	Description	Notes
border-top-width: *value*	Width of the top border	Where *value* is the width of the border in absolute or relative units, or defined with the keyword "thin", "medium", or "thick"
border-right-width: *value*	Width of the right border	
border-bottom-width: *value*	Width of the bottom border	
border-left-width: *value*	Width of the left border	
border-width: *top right bottom left*	Width of any or all of the borders	
border-top-color: *color*	Color of the top border	Where *color* is a color name or color value
border-right-color: *color*	Color of the right border	
border-bottom-color: *color*	Color of the bottom border	
border-left-color: *color*	Color of the left border	
border-color: *top right bottom left*	Color of any or all of the borders	
border-top-style: *type*	Style of top border	Where *type* is one of the nine border styles: solid, dashed, dotted, double, outset, inset, groove, ridge, or none
border-right-style: *type*	Style of right border	
border-bottom-style: *type*	Style of bottom border	
border-left-style: *type*	Style of left border	
border-style: *top right bottom left*	Style of any or all of the borders	

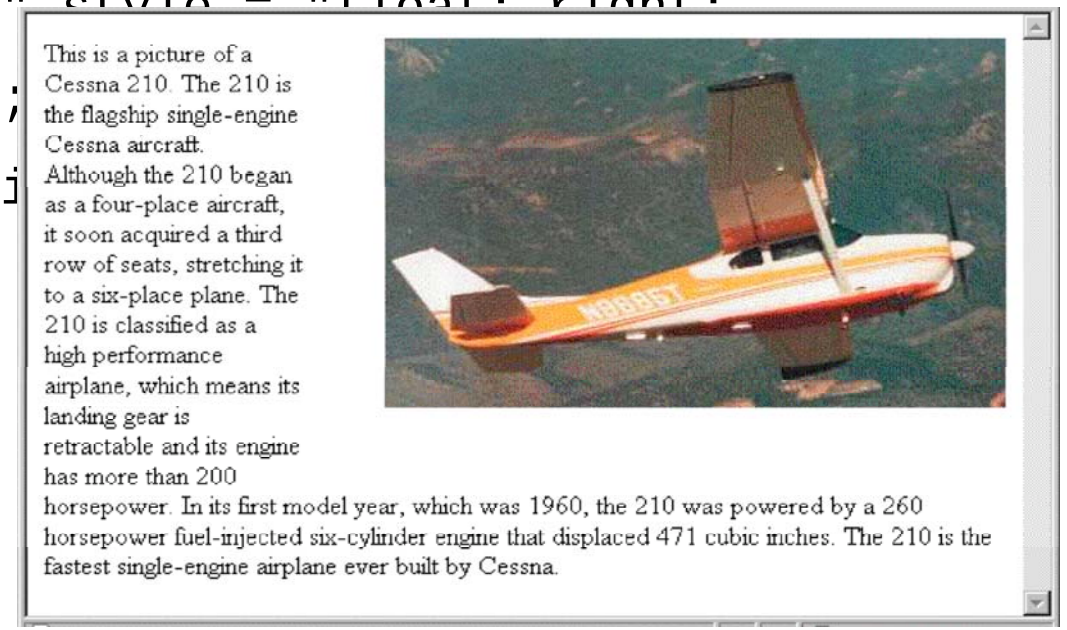
Border Style Types



The Box Model

- Margin – the space between the border of an element and its neighbor element
- The margins around an element can be set with `margin-left`, etc. - just assign them a length value

```
<img src = "c210.jpg " style = "float: right;
  margin-left: 0.35in;
  margin-bottom: 0.35i
```



The Box Model

- Padding – the distance between the content of an element and its border
  - Controlled by `padding`, `padding-left`, etc.

→ Example: `marpads.html`

- **Background Images**
- The `background-image` property

→ Can also specify `background-image`

  - Repetition can be controlled
    - `background-repeat` property
    - Possible values: `repeat` (default), `no-repeat`, `repeat-x`, or `repeat-y`
    - `background-position` property
      - Possible values: `top`, `center`, `bottom`, `left`, or `right`

The and tags

- One problem with the font properties is that they apply to whole elements, which are often too large
  - Solution: a new tag to define an element within a larger element -
  - Use to apply a document style sheet to its content

```
<style type = "text/css">
  bigred {font-size: 24pt; font-family: Ariel;
          color: red}
</style>
<p>
  Now is the
      <span class = "bigred"> best time </span> ever!
</p>
```

Positioning

- CSS2.1 has additional properties - one of the most important is *positioning*
  - *Normal Flow* - block formatting of block boxes, inline formatting of inline boxes, relative positioning of block or inline boxes
  - *Floats* - laid out according to normal flow, then shifted
  - *Absolute positioning* - box is removed entirely from normal flow
  - Values
    - static, relative, absolute, fixed
  - Offsets
    - top, right, left, bottom
  - See: http://www.w3.org/TR/CSS21/visuren.html

CSS2 Properties Positioning

- CSS-P was released by W3C in 1997
- Completely supported by IE9, FX3, and Chrome
- Each CSS box is laid out on the screen (or page) in one of the three ways: in its *normal* position, relative position or at an *absolute* position.
- This **`position`** property – value: **`static`** (i.e. “normal”), **`relative`**, **`absolute`**, or **`fixed`**; and the **`top`**, **`right`**, **`bottom`**, **`left`** offset properties – value: a length or percentage.

Absolute Positioning

- *Absolute Positioning*
  - The element is positioned relative to its first positioned (not static) ancestor

```
<p style = "position: absolute; left: 50px; top: 100px;">
```

→ SHOW `absPos.html`

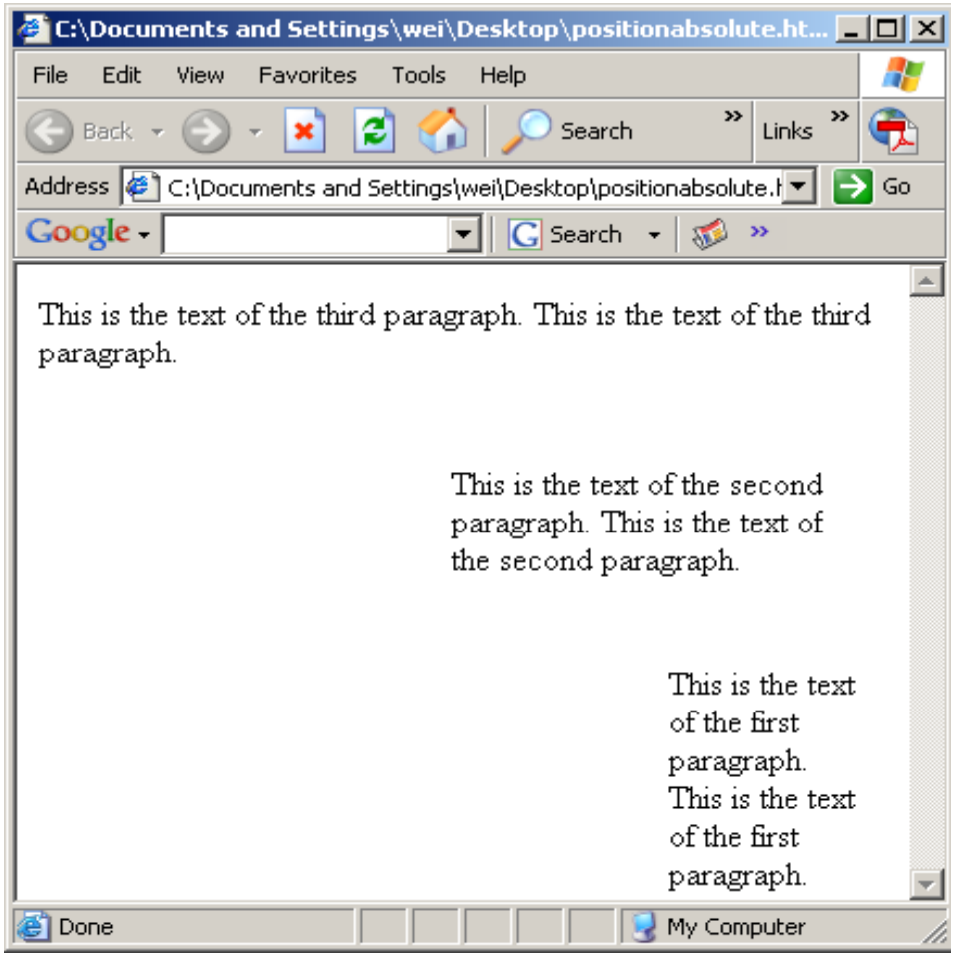
- If an element is nested inside another element and is absolutely positioned, the top and left properties are relative to the enclosing element

→SHOW `absPos2.html`

CSS2 Positioning Examples: Absolute

```
<html><head><style type="text/css">
  .one {position:absolute; top: 200px; left:300px}
  .two {position:absolute; top: 100px; left:200px}
  .three {position:static}
</style></head><body>
<p class="one">This is the text of the first paragraph.
  This is the text of the first paragraph. </p>
<p class="two">This is the text of the second paragraph.
  This is the text of the second paragraph. </p>
<p class="three">This is the text of the third paragraph.
  This is the text of the third paragraph. </p>
</body></html>
```

The Resulting Page



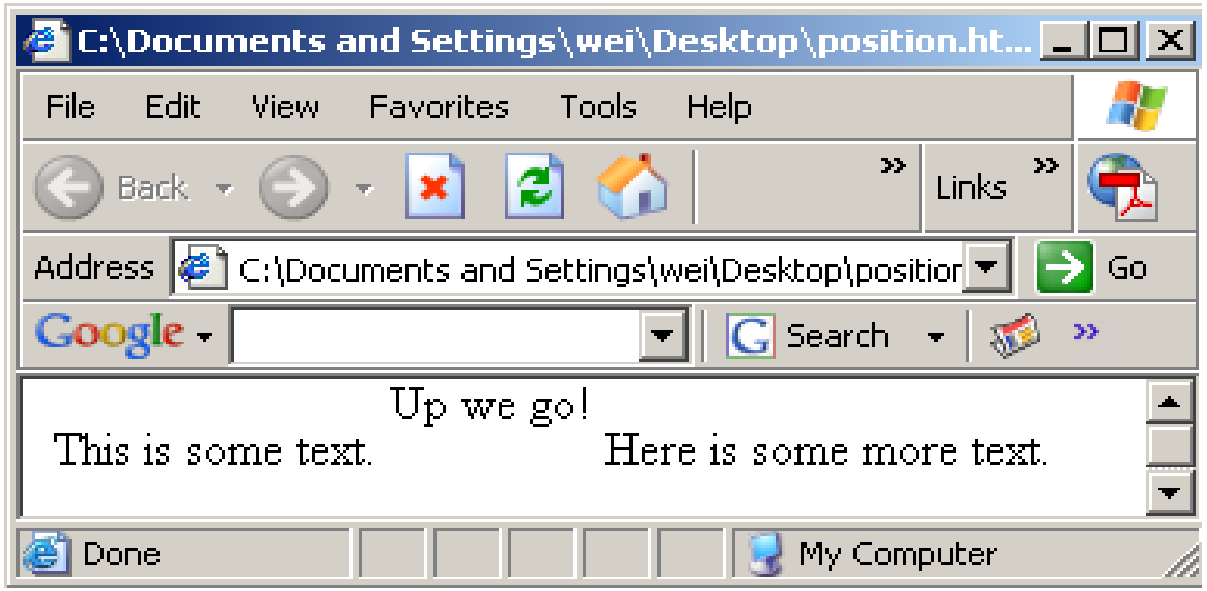
Relative Positioning

- *Relative Positioning*
  - If no `top` and `left` properties are specified, the element is placed exactly where it would have been placed if no `position` property were given
  - But it can be moved later using JavaScript

→ SHOW `relPos.html`

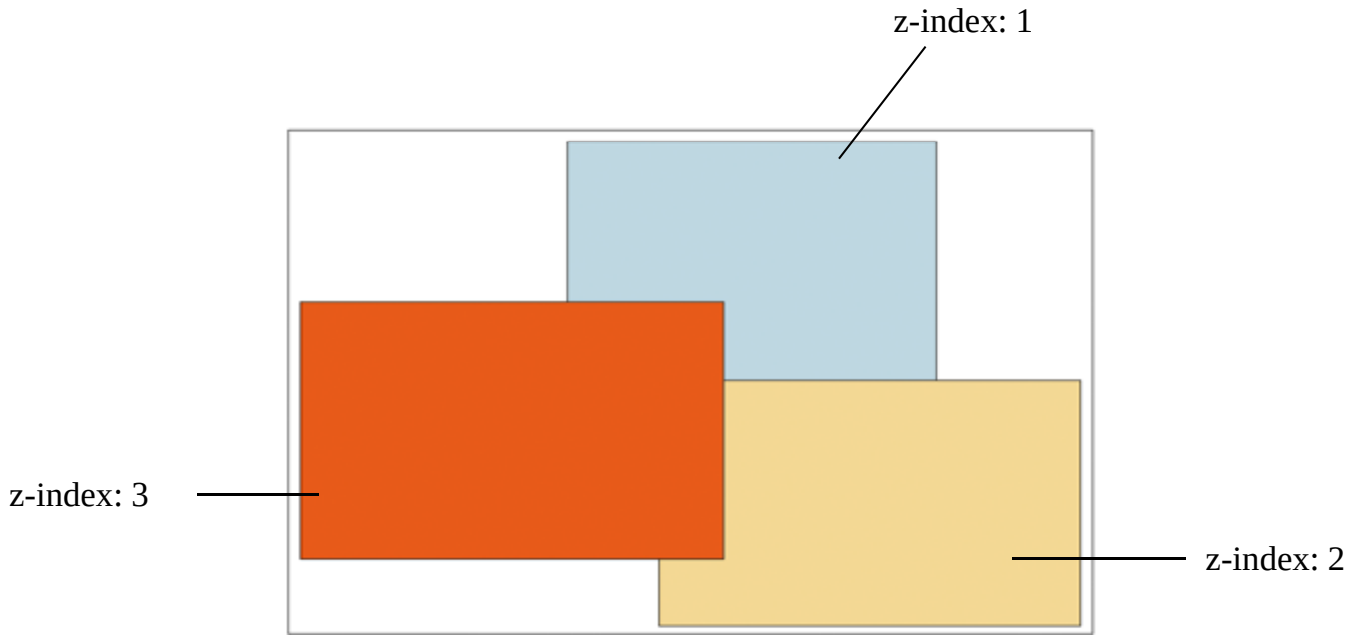
CSS2 Positioning Examples: Relative

```
<p> This is some text.

<span style="position:relative; top: -1em> Up we
  go!</span> Here is some more text.

</p>
```



Stacking Elements

- Specify stacking order with:
  - **z-index: value**



CSS2 Properties: 3D Layering

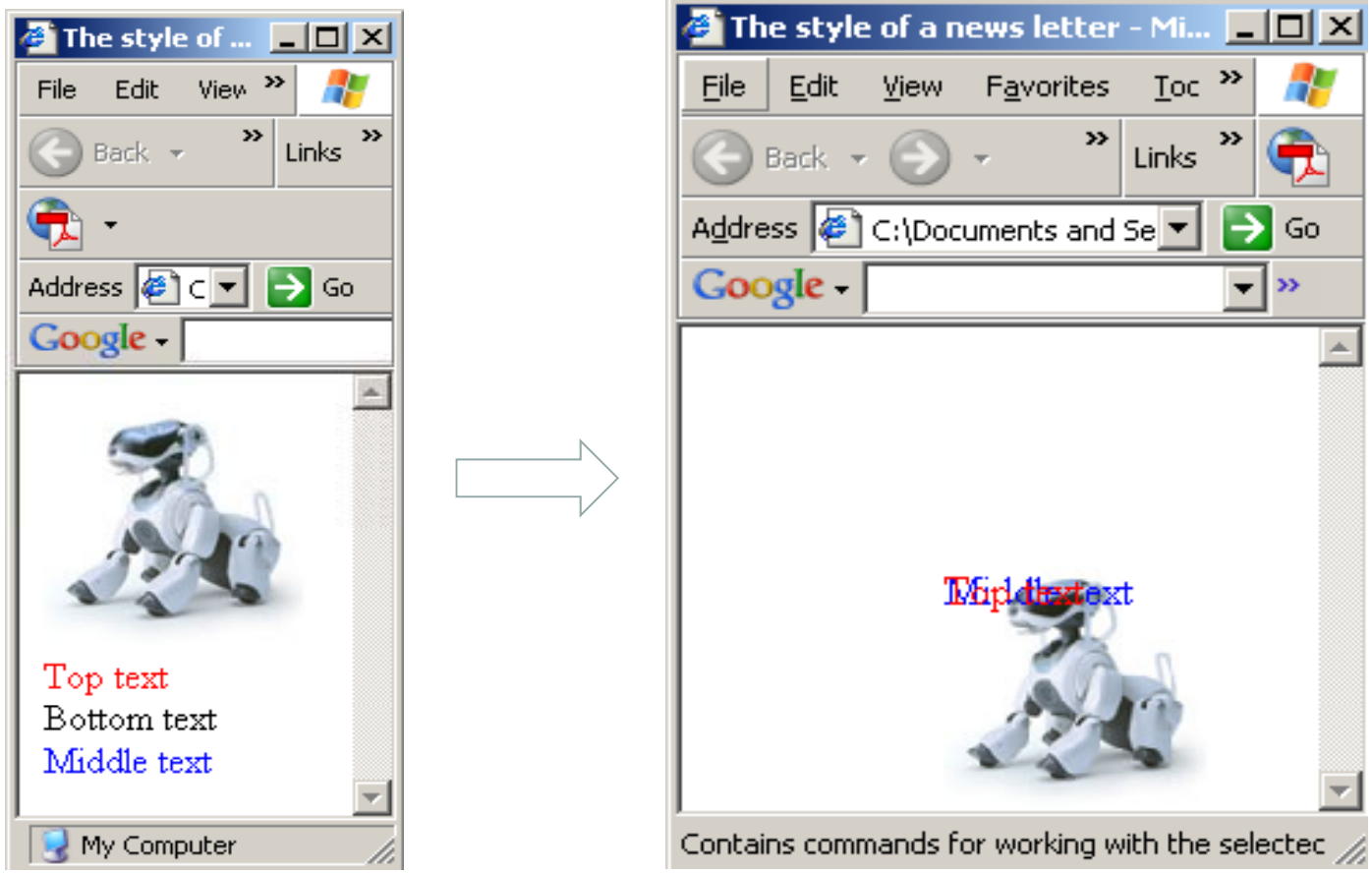
- In CSS2, each box has a position in three dimensions. In addition to their horizontal and vertical positions, boxes lie along a "z-axis" and are formatted one on top of the other.
- Positioning on the z-axis is controlled by the `z-index` property (0,1,2,3 …). The higher `z-index`ed objects are stacked above objects with lower z-index.
- Given:

```
.pile {position:absolute; top: 1in; left:1in;
       width:1in; height:1in}
```

How would this HTML snippet to be rendered?

```
<img class="pile" style="z-index:1" src="aibo.jpg" alt="AIBO" />
<div class="pile" style="z-index:3">Top text</div>
<div class="pile">Bottom text</div>
<div class="pile" style="z-index:2">Middle text</div>
```

Resulting page with v.s. without 3D layering

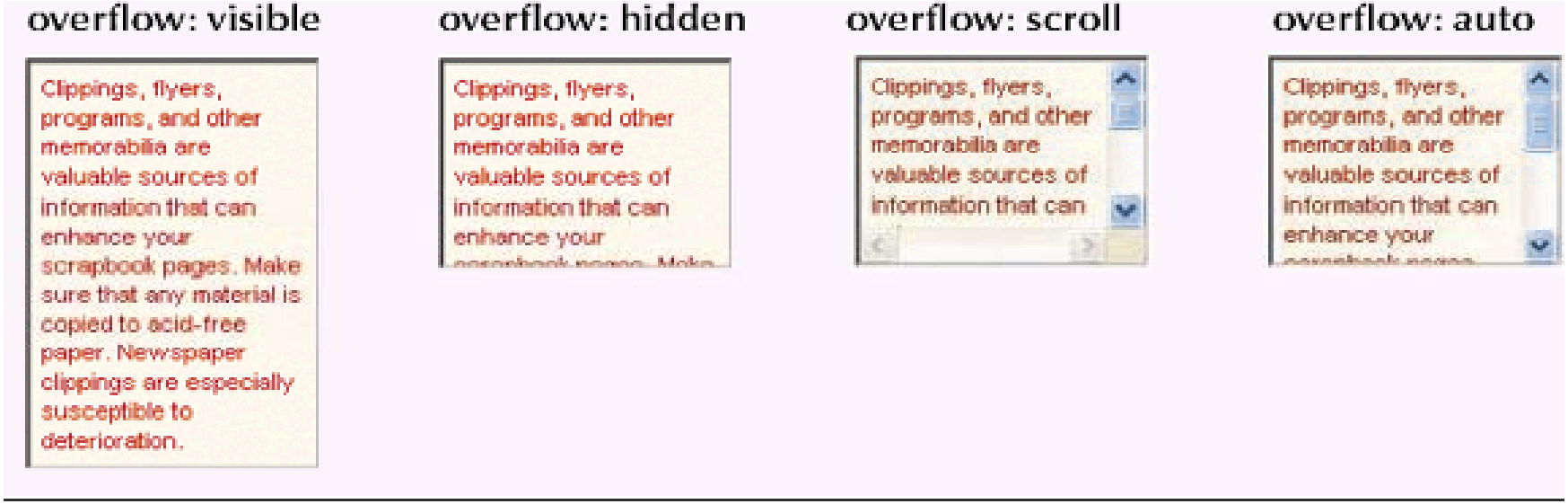


CSS2 Properties: Visibility, Overflow and Clipping

- The **`visibility`** property specifies whether the boxes generated by an element are rendered. Invisible boxes still affect layout.
- By default *visible*, a box is made invisible by setting its visibility to hidden:
  - `<p style="visibility: hidden"> ... </p>`
- Generally, the content of a box is confined within its edges. Sometimes, a box may overflow so that its content lies partly or entirely outside of the box. The **`overflow`** property specifies whether the content of a block is clipped (made invisible) when it overflows.
- Lastly, the **`clip`** property changes which part of a box is visible.

Working with overflow and clipping

- The overflow property syntax:
  - **overflow: type**



Setting the Display Style

Values of the display style

Display	Description
block	Display as a block-level element
inline	Display as an inline element
inline-block	Display as an inline element with some of the properties of a block (much like an inline image or frame)
inherit	Inherit the display property of the element's parent
list-item	Display as a list item
none	Do not display the element
run-in	Display as either an inline or block-level element depending on the context (CSS2)
table	Display as a block-level table (CSS2)

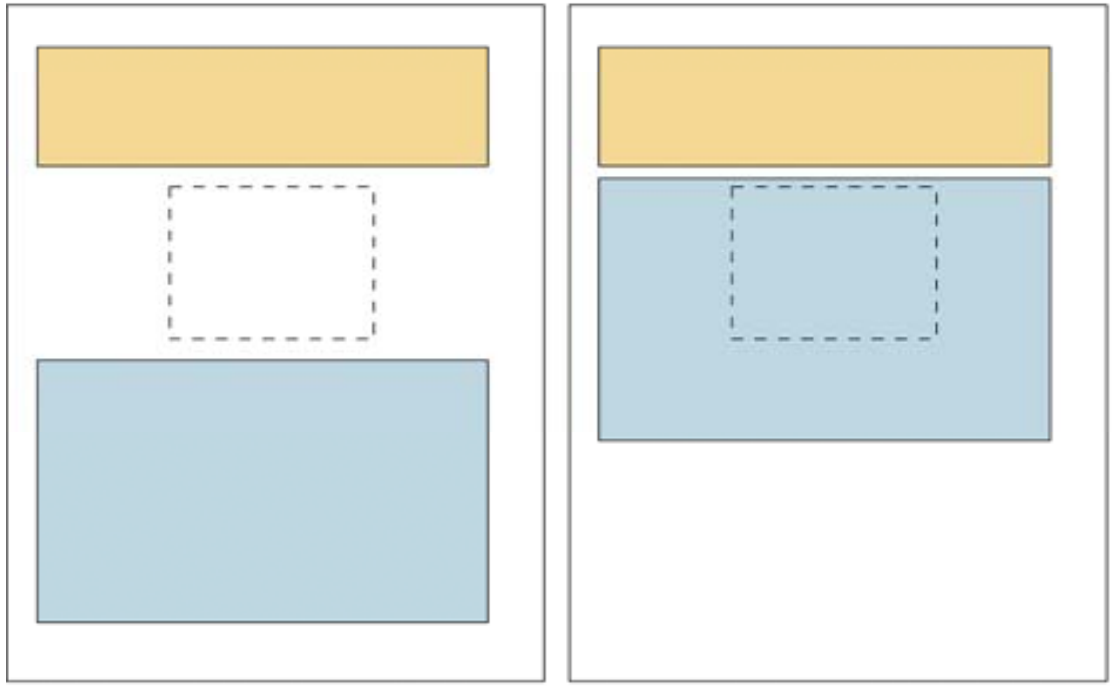
Setting the Display Style

Values of the display style

Display	Description
inline-table	Display as an inline table (CSS2)
table-caption	Treat as a table caption (CSS2)
table-cell	Treat as a table cell (CSS2)
table-column	Treat as a table column (CSS2)
table-column-group	Treat as a group of table columns (CSS2)
table-footer-group	Treat as a group of table footer rows (CSS2)
table-header-group	Treat as a group of table header rows (CSS2)
table-row	Treat as a table row (CSS2)
table-row-group	Treat as a group of table rows (CSS2)

Hiding Elements

- Two different styles that allow you to hide elements:
  - Display style
  - Visibility style



Visibility hidden

Object is hidden but still is part of the page flow

Display: none

Object is hidden and is removed from the page flow

Working with different media

Media Value	Used For
all	All output devices (the default)
aural	Speech and sound synthesizers
braille	Braille tactile feedback devices
embossed	Paged Braille printers
handheld	Small or handheld devices with small screens, monochrome graphics, and limited bandwidth
print	Printers
projection	Projectors
screen	Computer screens
tty	Fixed-width devices like teletype machines and terminals
tv	Television-type devices with low resolution, color, and limited scrollability

The @media Rule and media groups

- You can also specify the output media within a style sheet using:

  @media type {style declarations}

  where media is one of the supported media types and style declarations are the styles associated with that media type

- CSS2 uses media groups to describe basic facets of different media– and to differentiate between different types of media based on the ways they render content
  - Continuous or paged
  - Visual, aural, or tactile
  - Grid (for character grid devices) or bitmap
  - Interactive or static

Meida Groups

Media Types	Media Groups			
	continuous/paged	visual/aural/tactile	grid/bitmap	interactive/static
aural	continuous	aural	N/A	both
braille	continuous	tactile	grid	both
embossed	paged	tactile	grid	both
handheld	both	visual	both	both
print	paged	visual	bitmap	static
projection	paged	visual	bitmap	static
screen	continuous	visual	bitmap	both
tty	continuous	visual	grid	both
tv	both	visual, aural	bitmap	both

Using Print Styles

- You can specify the size of a page, margins, internal padding, etc. of the page box

CSS3 Online Resources

- CSS3 Information and Preview
  - http://www.css3.info
- CSS3 online rule generator
  - http://css3please.com/