CSP2103-4102: Markup Languages

Lecture 3: Cascading Style Sheets

Objectives

- Introducing Cascading Style Sheets
- Using Inline Styles
- Using Embedded Styles
- Using an External Style Sheet
- Understanding Cascading Order
- Working with Selectors

Objectives

- Using IDs and Classes
- Sizing Elements
- Floating an Element
- Working with the div Element
- Setting the Display Style
- Working with the Box Model
- Using Pseudo-Classes and Pseudo-Elements
- Applying a Style to a Web Site

Objectives

- Positioning Objects with CSS
- Working with Overflow and Clipping
- Stacking Elements
- Working with Different Media
- Hiding Elements
- Using Print Styles

Introducing Cascading Style Sheets

- **Style sheets** are files or forms that describe the layout and appearance of a document
- **Cascading Style Sheets**, or **CSS**, is a style sheet language used on the Web
  - CSS specifications are maintained by the World Wide Web Consortium (W3C)
  - Three versions of CSS exist: CSS1, CSS2, and CSS3

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

Applying a Style Sheet

- Three ways to apply a style to an HTML or XHTML document:
  - **Inline Styles**
  - **Embedded Styles**
  - **External Styles**

Using Inline Styles

- Inline styles are easy to use and interpret because they are applied directly to the elements they affect
- Inefficient if used repeatedly

```
<element style="style1: value1; style2: value2; style3: value3;…">
```

Using Embedded Styles

- You can embed style definitions in a document head using the following form
- Where style declarations are the declarations of the different styles to be applied to the document

```
<style>
style declarations
</style>
```

Using an External Style Sheet

- Because an embedded style sheet only applies to the content of the htm file, you need to place a style declaration in an **external style sheet** to apply to the headings in an entire Web site
- An **external style sheet** is a text file that contains style declarations
  - It can be linked to any page in the site, allowing the same style declaration to be applied to the entire site
  - Far more efficient the other methods

Using an External Style Sheet

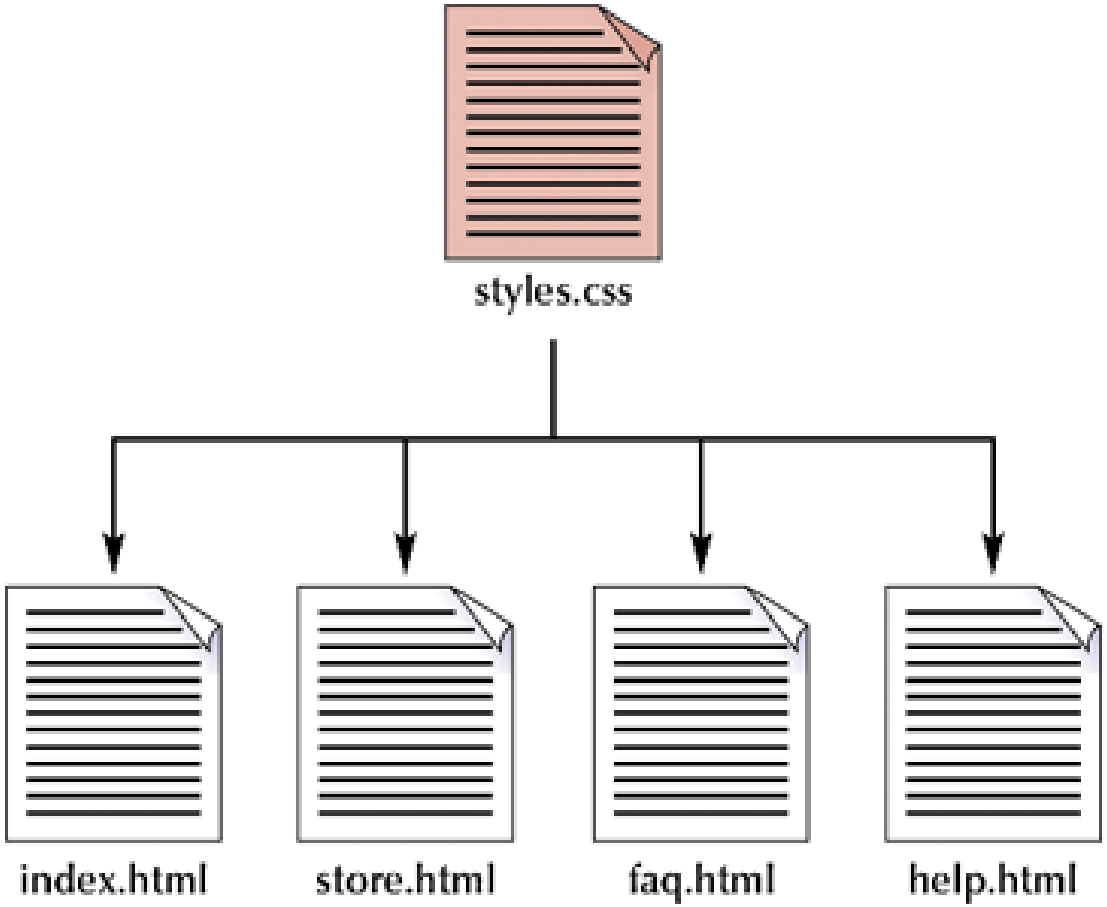
- You can add style comments as you develop an external style sheet
- You can import the content of one style sheet into another
- Use the link element to link a Web page to an external style sheet

```
<head>
<link rel="stylesheet" type="text/css"
href="mystyle.css" />
</head>
```

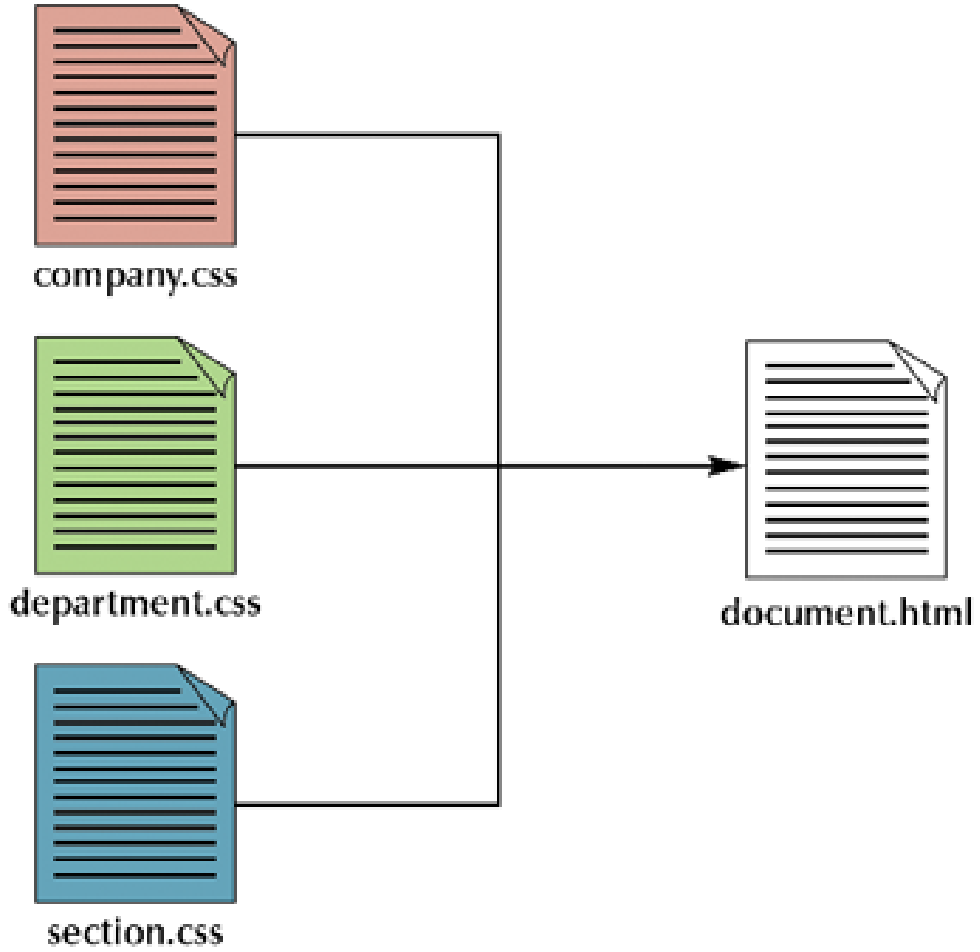
Understanding Cascading Order

- You can link a single style sheet to multiple documents in your Web site by using the link element or the @import element
- You can also link a single document to several style sheets
- A typical scenario has an organisation website using a single large style sheet which ALL pages must use
- Departments within the organisation might also use their own style sheet as an add-on to the central one

Applying a Single Style Sheet to Multiple Documents



Applying Multiple Sheets to a Single Document



Style Precedence and Inheritance

1. External style sheet
2. Embedded styles
3. Inline styles
4. If a style is not specified for an element, it inherits the style of its parent element; This is called **style inheritance.**

Working with Selectors

- CSS allows you to work with a wide variety of selectors to match different combinations of elements
- Use **contextual selectors** to apply a style based on the context in which an element is used
- Essentially, it is the rules for using CSS
- You can also create attribute selectors that select a given element when one of its attributes is used

Simple and Contextual Selectors

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*

Using IDs and Classes

- Use an id to distinguish something, like a paragraph, from the others in a document
  - For example, to identify a paragraph as “head”, use the code:

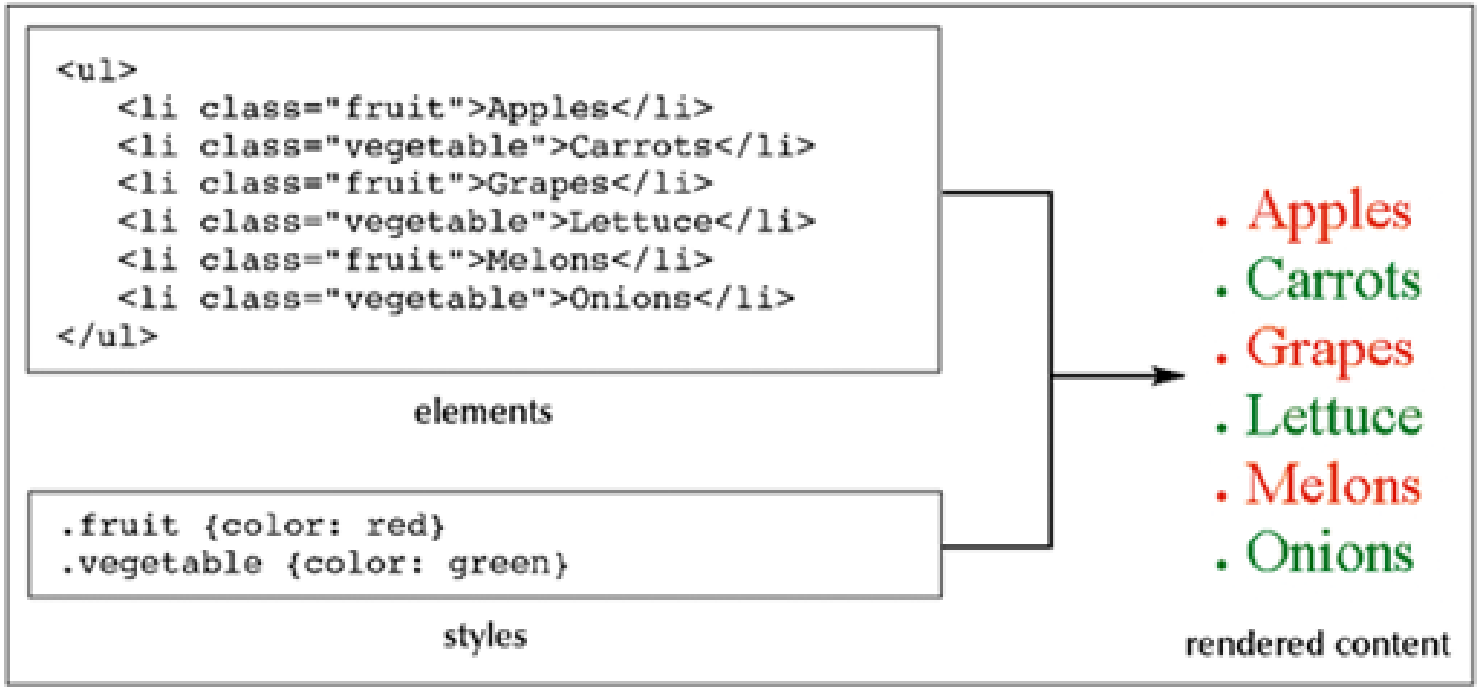
```
<p id=“head”>… </p>
```

Classes

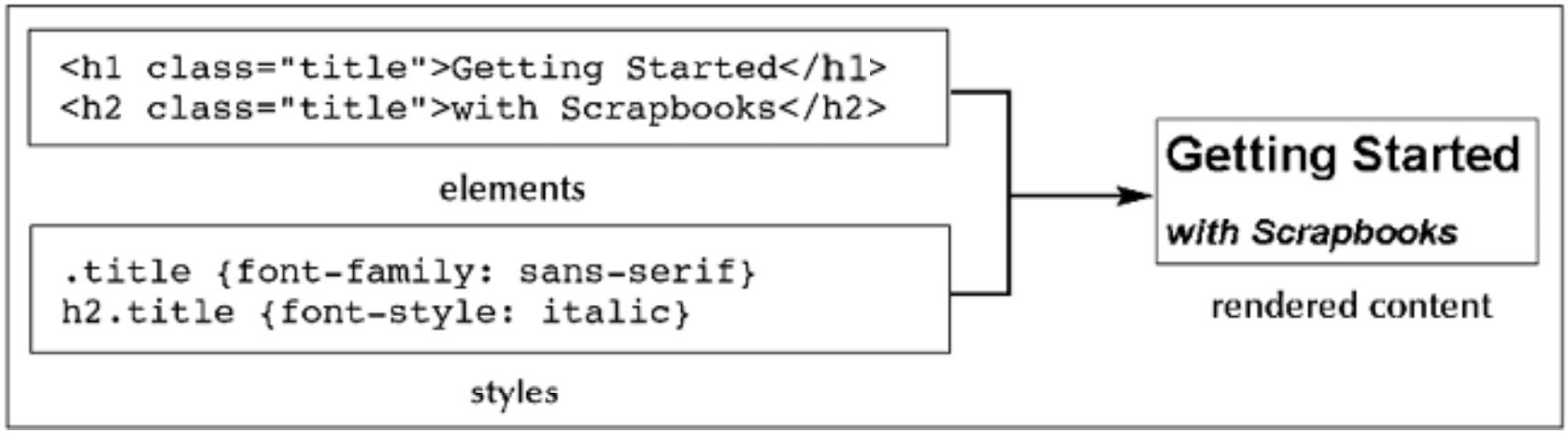
- HTML and XHTML require each id be unique– therefore an id value can only be used once in a document
- You can mark a group of elements with a common identifier using the class attribute

```
<element class="class"> … </element>
```

Applying a Style to a Class



Applying a Style to a Class and Element



Sizing Elements and Floating an Element

- You can define the width of columns in a columnar layout using: **`width: value`**
- You can use CSS to set an element's height using: **`height: value`**
- You can float a paragraph using: **`float: position`**
- As you may be seeing, what CSS gives you is word processor like control over the text, fonts and layouts in your html

Working with the div Element

- The **div element** is a generic block-level element
- Basically, is a logical **div**ider of content in the document

```
<div>
    content
</div>
```

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

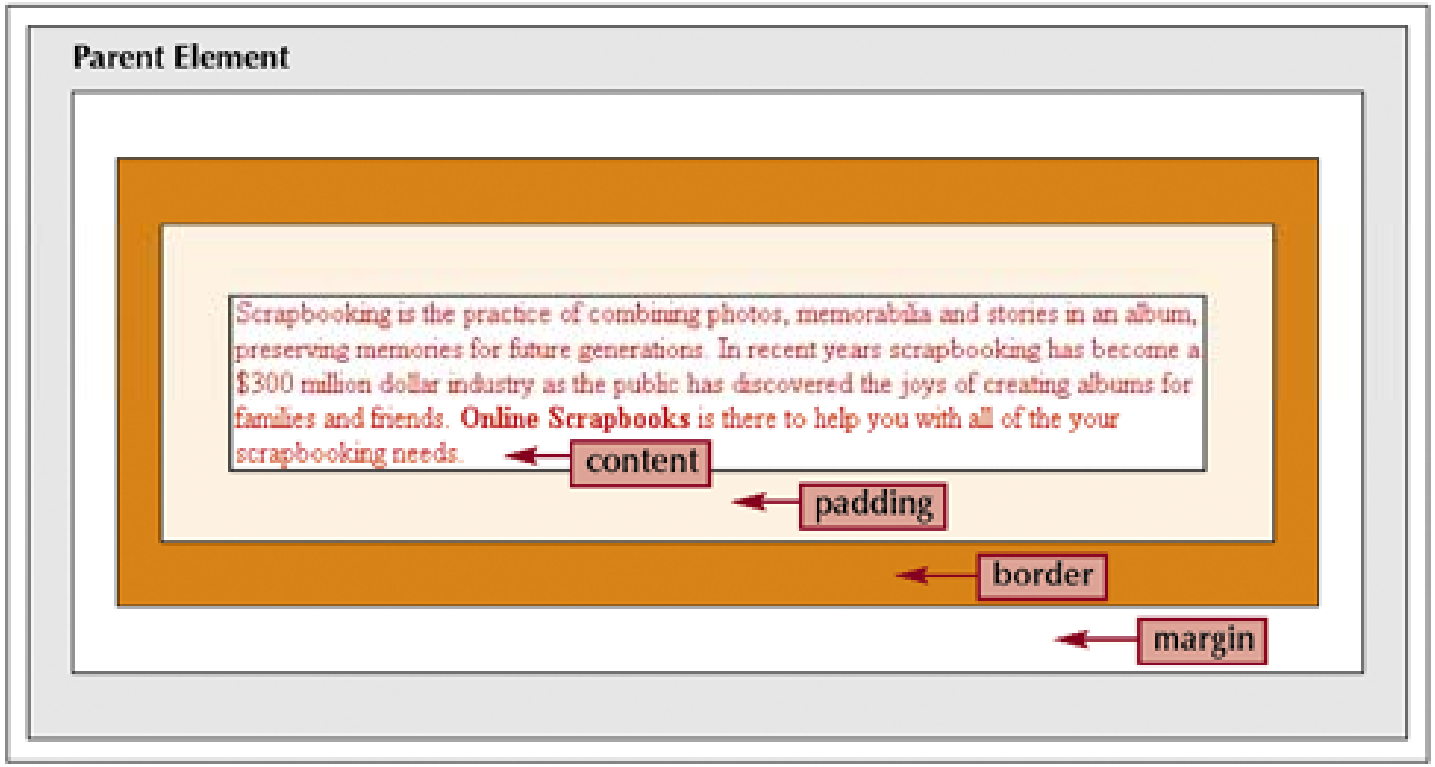
Values of the display style

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)

Working with the Box Model

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

The Box Model



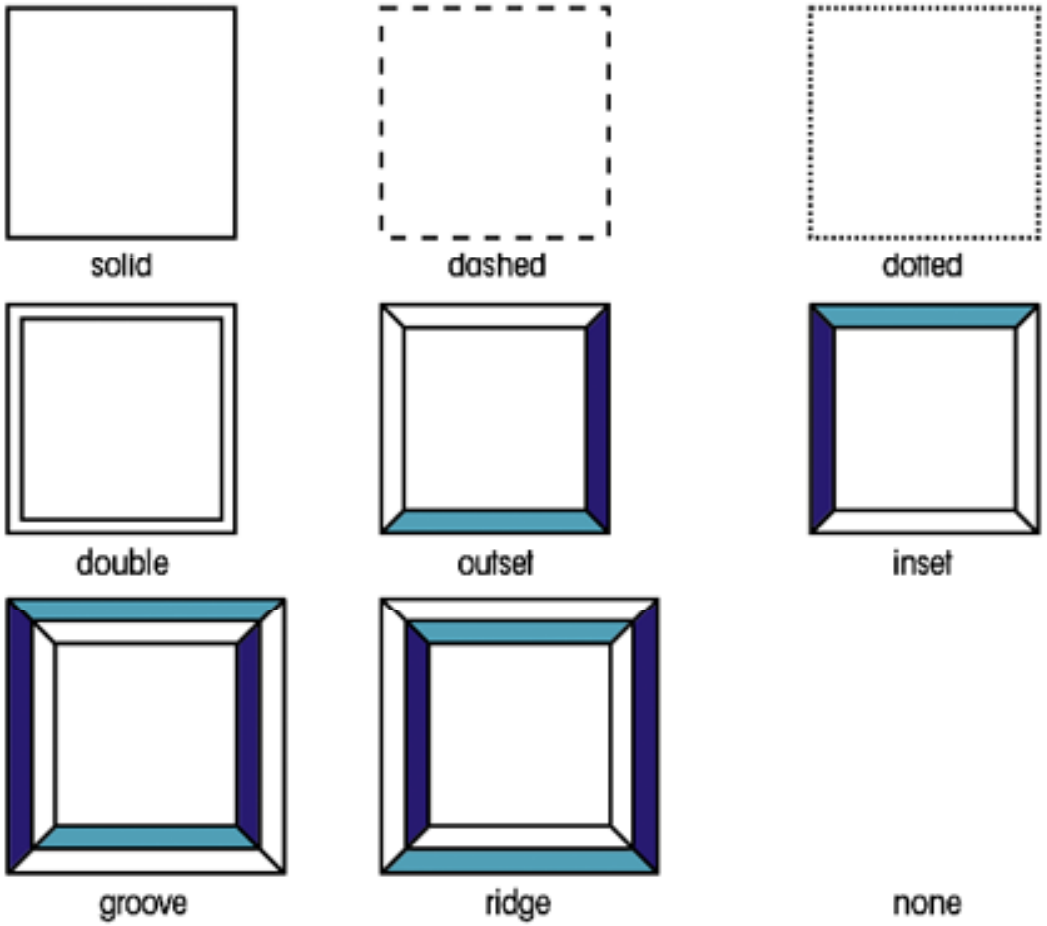
Working with the Box Model

- Styles to set padding are similar to styles to set margins:
  - **`padding-top: `*`value`**
  - **`padding-right: `*`value`**
  - **`padding-bottom: `*`value`**
  - **`padding-left: `*`value`**

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



Using Pseudo-Classes and Pseudo-Elements

- A **pseudo-class** is a classification of an element based on its status, position, or current use in the document

Pseudo-class	Description	Example
link	The link has not yet been visited by the user	a:link {color: red}
visited	The link has been visited by the user	a:visited {color: green}
active	The link is in the process of being activated by the user	a:active {color: yellow}
hover	The mouse cursor is hovering over the link (CSS2)	a:hover {color: blue}
focus	The element has received the focus of the keyboard or mouse cursor (CSS2)	input.focus {background-color: yellow}
first-child	The element is the first child of its parent (CSS2)	p:first-child {text-indent: 0}
lang	The element is in the specified language (CSS2)	q:lang(FR) {quotes: '<<' '>>'}

Using Pseudo-Classes and Pseudo-Elements

- **Rollover effects** can be created using pseudo-classes
- **Pseudo-elements** are elements based on information about an element's content, use or position

Pseudo-element	Description	Example
first-letter	The first letter of the element text	p:first-letter {font-size: 14pt}
first-line	The first line of the element text	p:first-line {text-transform: uppercase}
before	Content to be placed directly before the element (CSS2)	p:before {content: "Special!"}
after	Content to be placed directly after the element (CSS2)	p:after {content: "eof"}

Positioning Objects with CSS

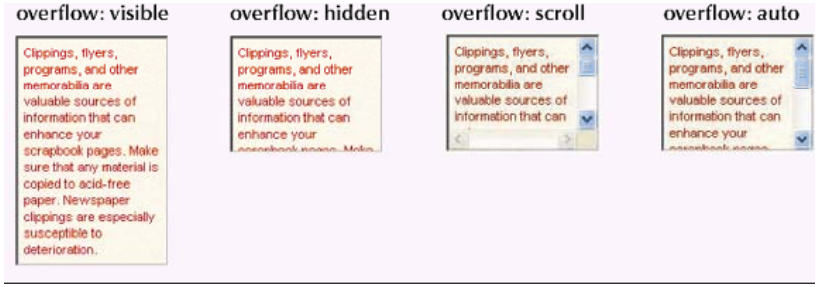
- The different positioning styles in the original CSS1 specifications were known as CSS-Positioning or CSS-P
- To place an element at a specific position on a page use:

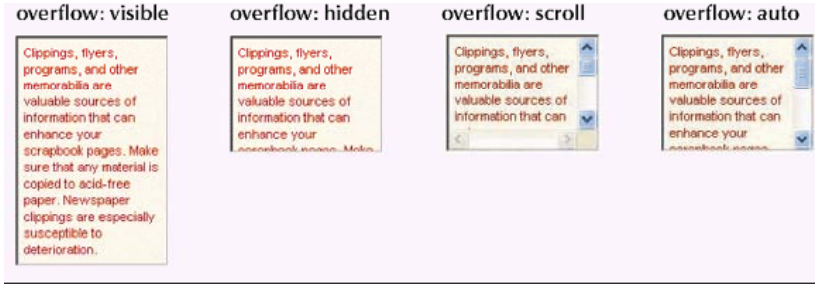
```
position: type; top: value; right: value;
bottom: value; left: value;
```

Working with Overflow and Clipping

- The overflow property syntax:

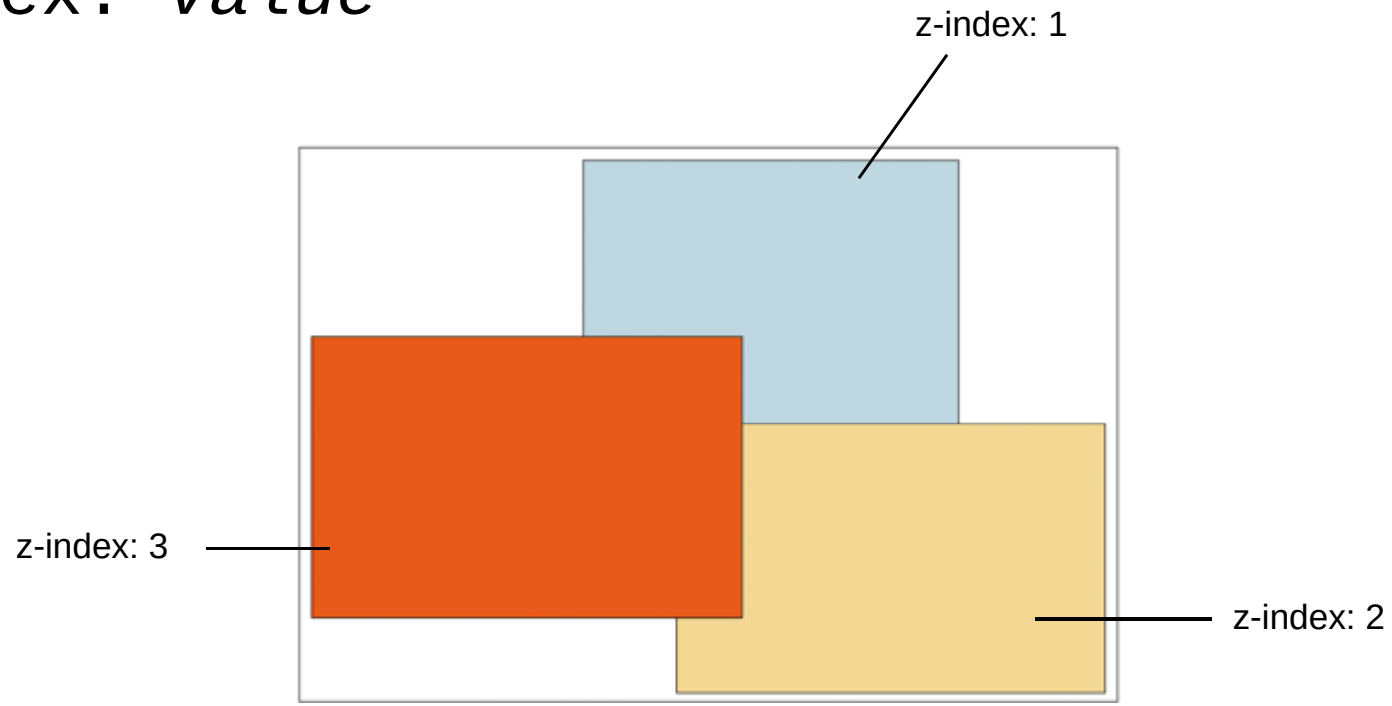
  `overflow: type`





Stacking Elements

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



JavaScript and Stacked Elements

- The beauty of stacked elements is that you can use JavaScript to dynamically change the z-index of a given element
- For example, you could have a series of div elements which are stacked on top of each other
- You could use a JavaScript rollover to change the z-index of a given div, bringing it to the top of the stack (making it visible) instantly
- This is how many drop-down menus are done

Working with Different Media

- Specify output styles for particular devices in the media attribute of the link and style elements
- Useful, but could be considered overkill unless there was an absolute need

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

- 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

Media Groups

- 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

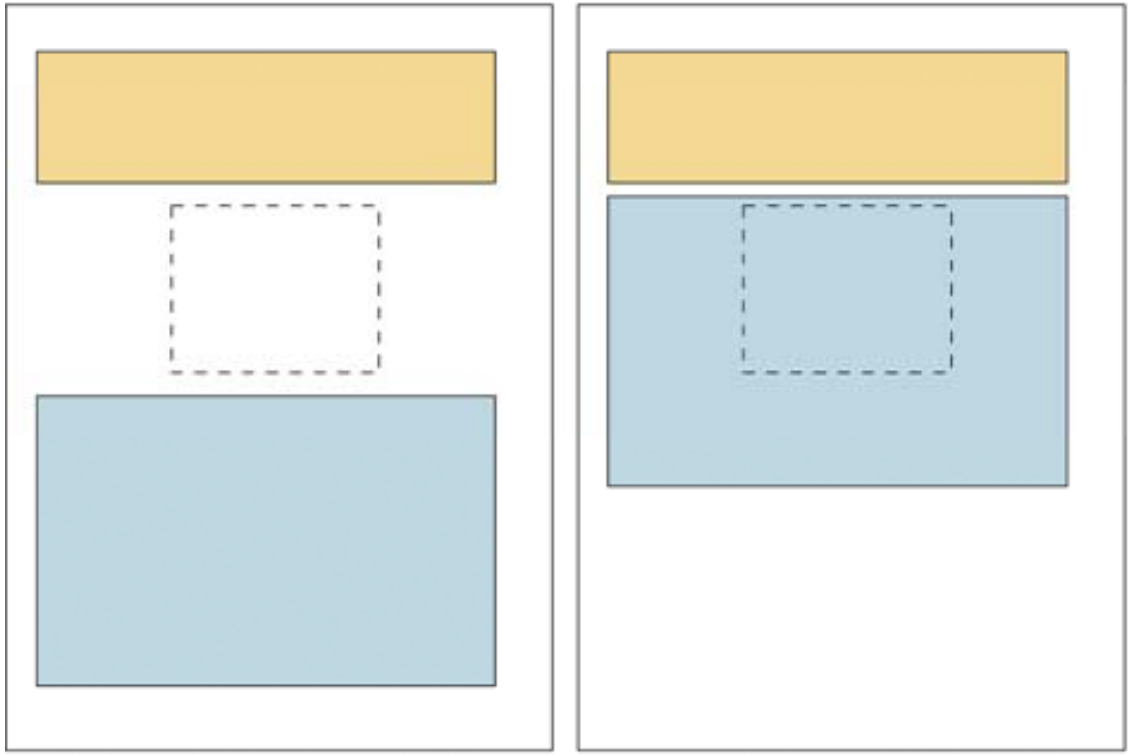
Media Groups

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

Hiding Elements

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

Comparing the visibility and display styles



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

Using Print Styles

- You can specify the size of a page, margins, internal padding, etc. of the page box
- This is very useful for content that needs to be printed to actual paper
- Typically, you would have a link to ‘Printer Version’, then on that page have your CSS geared to page printing

Conclusion

- CSS has gone from being considered an added extra for web pages to being essential
- XHTML has seen many of the old HTML formatting elements deprecated, as it is now expected that CSS be used even for basic formatting
- External style sheets can take a while to develop, but once completed make page development very easy, and above all, consistent