3. Markup Languages and Formatting

a. HTML

b. XHTML

c. CSS (Cascading Style Sheets)

2.1

HTML

What is an HTML File?

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is *not* a computer programming language.
  - It is simply a set of markup codes that structure and style text and graphics.
  - Learning HTML requires learning these markup tags.
- An HTML file can be created using a simple text editor

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage.
<b>This text is bold</b>
</body>
</html>
```

- HTML Editors
  - WYSIWYG editor like FrontPage, Macromedia HomeSite, or Adobe PageMill
  - To be a skillful Web developer, use a plain text editor!

HTML Tags

- HTML tags are used to mark-up HTML elements
- HTML tags normally come in pairs like and
- The text between the start and end tags is the element content
- HTML tags are not case sensitive, means the same as <B>
  - W3C recommends lowercase tags in their HTML 4 recommendation, and XHTML (the next generation HTML) demands lowercase tags
- Tag Attributes
  - Tags can have attributes. Attributes can provide additional information about the HTML elements on your page
    - `<body bgcolor="red">`
  - Attributes always come in name/value pairs like this: `name="value"`
  - Attribute values should always be enclosed in quotes

General Structure of an HTML File

```
<html>
      <head>
          <title>Title of page</title>
      </head>
    <body>
       This is my first homepage.
        <b>This text is bold</b>
    </body>
</html>
```

HTML Container and Empty Tags

- There are two basic types of tags that are used in a HTML document (web page):
  - Container tags.
  - Empty tags
- Container tags are tags that enclose, between their start and end tags, other HTML tags or text
  - <html>… </html>, <body>…</body>, …
- EMPTY tags do not contain any text.
  - They are used simply as markers (and in some cases are used for whatever is contained in their attributes).
  - EMPTY elements are not permitted to have an end-tag.
    - Thus `<img src="blahblah.gif"></img>` is illegal.
- Example EMPTY elements are
  - **hr** Horizonal rule
  - **br** Line break
  - **img** Inline image
  - **input** Input
- Container and EMPTY tags can have attributes

The <head> tag

- The head element contains general information (meta-information) about a document
  - Using this is optional, but recommended.
- Head Tags
  - `<title>`: defines the document title
  - `<base>`: defines a base URL for all the links
    - `<base href="http://www.w3schools.com/images/" />`
  - `<link>`: defines a resource reference
    - `<link rel="stylesheet" type="text/css" href="theme.css" />`
  - `<meta>`: defines meta information about your page, such as descriptions and keywords for search engines and refresh rates

… The <head> tag

- metadata is always passed as name/value pairs
- `<meta name="keywords" content="HTML, DHTML, CSS, XML, XHTML, JavaScript, VBScript" />`
- `<meta http-equiv="refresh" content="5" />`
- `<meta http-equiv="Content-Type" content="text/html; charset=windows-1256">`
- `ISO-8859-6 (Arabic)`

Attribute	Value	Description
http-equiv	content-type expires refresh set-cookie	Connects the content attribute to an HTTP header
name	author description keywords generator revised *others*	Connects the content attribute to a name

Basic HTML Tags

- Headings
  - Headings are defined with the <h1> to <h6> tags
- Paragraphs
  - Paragraphs are defined with the <p> tag
  - HTML automatically adds an extra blank line before and after a paragraph
- Line Breaks
  - The <br> tag is used when you want to end a line, but don't want to start a new paragraph
- Horizontal Rule: the <hr> tag
- Comments in HTML
  - `<!-- This is a comment -->`
- HTML will truncate the spaces in your text. Any number of spaces count as one

HTML Text Formatting

Tag	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<blockquote>	Defines a long quotation
<code>	Defines computer code text
<pre>	Defines preformatted text

HTML Character Entities

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	quotation mark	&quot;	&#34;
'	apostrophe		&#39;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
×	multiplication	&times;	&#215;
÷	division	&divide;	&#247;

HTML Fonts

- The <font> tag in HTML is deprecated.
  - It is supposed to be removed in a future version of HTML.
- Even if a lot of people are using it, you should try to avoid it, and use styles instead

HTML Links

- The Anchor Tag
  - the `href` Attribute
    - `<a href="http://www.w3schools.com/">`Visit W3Schools!`</a>`
    - always add a trailing slash to subfolder references
  - the `target` attribute defines where the linked document will be opened
    - `<a href="http://www.w3schools.com/" target="_blank">`Visit W3Schools!`</a>`
  - the `name` Attribute
    - `<a name="tips">`Useful Tips Section`</a>`
    - `<a href="http://www.w3schools.com/html_links.asp#tips">`Jump to the Useful Tips Section`</a>`
    - a hyperlink to the Useful Tips Section from WITHIN the file "html_links.asp" will look like this
      - `<a href="#tips">`Jump to the Useful Tips Section`</a>`

… HTML Links

- A `mailto` link

```
<html>

<body>

<p>
This is a mail link:
<a href="mailto:someone@microsoft.com?subject=Hello%20again">
Send Mail</a>
</p>

</body>
</html>
```

Frames

- With frames, you can display more than one HTML document in the same browser window
  - Each HTML document is called a frame, and each frame is independent of the others
- The disadvantages of using frames are:
  - The web developer must keep track of more HTML documents
  - It is difficult to print the entire page
- The Frameset Tag
  - The <frameset> tag defines how to divide the window into frames
  - Each frameset defines a set of rows or columns
  - The values of the rows/columns indicate the amount of screen area each row/column will occupy
- The Frame Tag
  - The <frame> tag defines what HTML document to put into each frame
- Useful Tips
  - If a frame has visible borders, the user can resize it by dragging the border. To prevent a user from doing this, you can add noresize="noresize" to the <frame> tag.
  - Add the <noframes> tag for browsers that do not support frames

… Frames

Mixed frameset

```
<html>

<frameset rows="50%,50%">

<frame src="tryhtml_frame_a.htm">

<frameset cols="25%,75%">
<frame src="tryhtml_frame_b.htm">
<frame src="tryhtml_frame_c.htm">
</frameset>

</frameset>

</html>
```

Navigation frame

```
<html>

<frameset cols="120,*">

<frame
src="tryhtml_contents.htm">
<frame src="tryhtml_frame_a.htm"
name="showframe">

</frameset>

</html>
```

```
<a href ="tryhtml_frame_a.htm"
target ="showframe">Frame a</a><br>
```

Inline frame

- The iframe element creates an inline frame that contains another document (a frame inside an HTML page)

```
<iframe src ="/default.htm"> </iframe>
```

Attribute	Value	Description
align	left right top middle bottom	Specifies how to align the iframe according to the surrounding text
frameborder	1 0	Specifies whether or not to display a frame border
height	pixels %	Defines the height of the iframe
longdesc	URL	A URL to a long description of the frame contents
marginheight	pixels	Defines the top and bottom margins of the iframe
marginwidth	pixels	Defines the left and right margins of the iframe
name	frame_name	Specifies a unique name of the iframe (to use in scripts)
scrolling	yes no auto	Define scroll bars
src	URL	The URL of the document to show in the iframe
width	pixels %	Defines the width of the iframe

Tables

- One very common practice with HTML, is to use HTML tables to format the layout of an HTML page
- Tables are defined with the <table> tag
- A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag)
- A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr> </table>
```

Table Tags

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer

```
<table border="1" cellpadding="10">
<tr>
  <th>Name</th>
  <th colspan="2">Telephone</th>
</tr>
<tr>
  <td>Bill Gates</td>
  <td>555 77 854</td>
  <td>555 77 855</td>
</tr>
</table>
```

The <table> tag attributes

Attribute	Value	Description
align	left center right	Aligns the table. Deprecated. Use styles instead.
bgcolor	rgb(x,x,x) #xxxxxx colorname	Specifies the background color of the table. Deprecated. Use styles instead.
border	pixels	Specifies the border width. **Tip:** Set border="0" to display tables with no borders!
cellpadding	pixels %	Specifies the space between the cell walls and contents
cellspacing	pixels %	Specifies the space between cells
frame	void above below hsides lhs rhs vsides box border	Specifies how the outer borders should be displayed. **Note:** Must be used in conjunction with the "border" attribute!
rules	none groups rows cols all	Specifies the horizontal/vertical divider lines. **Note:** Must be used in conjunction with the "border" attribute!
width	% pixels	Specifies the width of the table

The <td> tag attributes

Attribute	Value	Description
abbr	abbr_text	Specifies an abbreviated version of the content in a cell
align	left right center justify char	Specifies the horizontal alignment of cell content
bgcolor	rgb(x,x,x) #xxxxxx colorname	Specifies the background color of the table cell. Deprecated. Use styles instead.
colspan	number	Indicates the number of columns this cell should span
height	pixels	Specifies the height of the table cell. Deprecated. Use styles instead.
nowrap	nowrap	Whether to disable or enable automatic text wrapping in this cell. Deprecated. Use styles instead.
rowspan	number	Indicates the number of rows this cell should span
valign	top middle bottom baseline	Specifies the vertical alignment of cell content
width	pixels %	Specifies the width of the table cell. Deprecated. Use styles instead.

Lists

- Unordered Lists
  - The list items are marked with bullets (typically small black circles)
  - An unordered list starts with the `<ul>` tag. Each list item starts with the `<li>` tag
  - `<ul> <li>Coffee</li> <li>Milk</li> </ul>`
- Ordered Lists
  - The list items are marked with numbers.
  - An ordered list starts with the `<ol>` tag. Each list item starts with the `<li>` tag.
- Definition Lists
  - This is a list of terms and explanation of the terms
  - A definition list starts with the `<dl>` tag. Each definition-list term starts with the `<dt>` tag. Each definition-list definition starts with the `<dd>` tag.

Forms

- A form is an area that can contain form elements
  - Form elements are elements that allow the user to enter information (like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.) in a form
  - `<form> <input> </input> </form>`
- Input
  - The most used form tag is the <input> tag. The type of input is specified with the type attribute.
  - Text Fields
  - Radio Buttons
  - Checkboxes
  - Drop Lists

... Forms

```
<form>
First name: <input type="text“ name="firstname">
<br>
Last name: <input type="text" name="lastname">
<input type="radio" name="sex" value="male" checked> Male
<br>
<input type="radio" name="sex" value="female"> Female
<input type="checkbox" name="bike"> I have a bike
<br>
<input type="checkbox" name="car"> I have a car
<select name="cars">
  <option value="volvo">Volvo
  <option value="saab">Saab
  <option value="fiat">Fiat
  <option value="audi">Audi
</select>
<textarea rows="10" cols="30">
</textarea>
</form>
```

... Forms

- The Form's Action Attribute and the Submit Button
  - When the user clicks on the "Submit" button, the content of the form is sent to another file.
  - The form's action attribute defines the name of the file to send the content to
  - `<input type="submit" value="Send">`

... Forms

```
<form action="MAILTO:someone@w3schools.com" method="post"
enctype="text/plain">

<h3>This form sends an e-mail to W3Schools.</h3>
Name:<br>
<input type="text" name="name"
value="yourname" size="20">
<br>
Mail:<br>
<input type="text" name="mail"
value="yourmail" size="20">
<br>
Comment:<br>
<input type="text" name="comment"
value="yourcomment" size="40">
<br><br>
<input type="submit" value="Send">
<input type="reset" value="Reset">

</form>
```

Images

- images are defined with the `<img>` tag
  - `<img src="boat.gif" alt="Big Boat">`
  - optional attributes

align	top bottom middle left right	Specifies how to align the image according to surrounding text. Deprecated. Use styles instead
border	pixels	Defines a border around an image. Deprecated. Use styles instead
height	pixels %	Defines the height of an image
hspace	pixels	Defines white space on the left and right side of the image. Deprecated. Use styles instead
ismap	URL	Defines the image as a server-side image map
longdesc	URL	A URL to a document that contains a long description of the image
usemap	URL	Defines the image as a client-side image map. Look at the <map> and <area> tags to figure out how it works
vspace	pixels	Defines white space on the top and bottom of the image. Deprecated. Use styles instead
width	pixels %	Sets the width of an image

… Images

Let the image float

```
<p>
<img src ="/images/xhtml.gif"
align ="left" width="100" height="50">
A paragraph with an image. The align attribute of the
image is set to "left". The image will float to the
left of this text.
</p>
```

Make a hyperlink of an image

```
<p>
You can also use an image as a link:
<a href="lastpage.htm">
<img border="0" src="buttonnext.gif" width="65" height="38">
</a>
</p>
```

Image Maps

- An imagemap allows you to create links to different URLs according to where you click on the image
- Imagemaps are useful for creating links on maps, diagrams, fancy buttons, etc.
- The map file defines the areas of the image and the URLs that correlate to each different areas.
- There are two types of image maps:
  - *Client-side.* When a user activates a region of a client-side image map with a mouse, the pixel coordinates are interpreted by the user agent. The user agent selects a link that was specified for the activated region and follows it.
  - *Server-side.* When a user activates a region of a server-side image map with a mouse, the pixel coordinates of the click are sent to the server-side agent specified by the `href` attribute of the A element. The server-side agent interprets the coordinates and performs some action.

An Image Map Example

```
<p>
Click on one of the planets to watch it closer:
</p>

<img src="planets.gif"
width="145" height="126"
usemap="#planetmap">

<map id="planetmap" name="planetmap">

<area shape="rect"
coords="0,0,82,126"
alt="Sun"
href="sun.htm">

<area shape="circle"
coords="90,58,3"
alt="Mercury"
href="mercur.htm">

<area shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm">

</map>
```

Backgrounds

- The background can be a color or an image
  - `<body bgcolor="#000000">`
  - `<body bgcolor="rgb(0,0,0)">`
  - `<body bgcolor="black">`
  - `<body background="clouds.gif">`
  - `<body background="http://www.w3schools.com/clouds.gif">`
  - the background image will increase the loading time
  - The bgcolor, background, and the text attributes in the <body> tag are deprecated in the latest versions of HTML (HTML 4 and XHTML)
  - In future versions of HTML, style sheets (CSS) will be used to define the layout and display properties of HTML elements

Reference

- A useful link with examples and other resources:
  - http://www.w3schools.com/html/
  - http://www.boutell.com/mapedit

3.2

XHTML

What is XHTML?

- XHTML stands for E**X**tensible **H**yperText **M**arkup **L**anguage
- XHTML is the next generation of HTML
- XHTML is aimed to replace HTML
- XHTML is almost identical to HTML 4.01
- XHTML is a stricter and cleaner version of HTML
- XHTML is a reformulation of HTML into a language that conforms to the XML 1.0 Recommendation
- XHTML Family document types are all XML-based, and ultimately are designed to work in conjunction with XML-based user agents

Why XHTML?

- XHTML is a combination of HTML and XML
  - XML is a markup language where everything has to be marked up correctly, which results in "well-formed" documents
- XHTML consists of all the elements in HTML 4.01 combined with the syntax of XML
- We have reached a point where many pages on the WWW contain "bad" HTML
- XHTML pages can be read by all XML enabled devices
  - An interim solution before the rest of the world upgrades to XML

… Why XHTML?

- Separation of concern between document structuring and document formatting
- Remove formatting information from HTML
  - Deprecate html tags and attributes for display and formatting
  - Make place for CSS!
- Conformance with XML syntax

Differences Between XHTML and HTML

- The Most Important Differences:
  - XHTML elements must be properly nested
  - XHTML documents must be well-formed
  - Tag names must be in lowercase
  - All XHTML elements must be closed

Elements Must Be Properly Nested

- In HTML some elements can be improperly nested within each other like this:
  - `<b><i>This text is bold and italic</b></i>`
  - In XHTML all elements must be properly nested within each other like this:
  - `<b><i>This text is bold and italic</i></b>`
- A common mistake in nested lists, is to forget that the inside list must be within an li element

Documents Must Be Well-formed

- All XHTML elements must be nested within the `<html>` root element
  - All other elements can have sub (children) elements. Sub elements must be in pairs and correctly nested within their parent element.
  - The basic document structure is:

```
<html>
<head> ... </head>
<body> ... </body>
</html>
```

Tag Names Must Be in Lower Case

- This is because XHTML documents are XML applications
- XML is case-sensitive
- Tags like `<br>` and `<BR>` are interpreted as different tags

All XHTML Elements Must Be Closed

- Non-empty elements must have an end tag
- Empty Elements Must also Be Closed
  - Empty elements must either have an end tag or the start tag must end with />

```
This is a break<br />

Here comes a horizontal rule:<hr />

Here's an image <img src="happy.gif" alt="Happy face" />
```

XHTML Syntax

- Writing XHTML demands a clean HTML syntax
- Some more XHTML Syntax Rules:
  - Attribute names must be in *lower case*
  - Attribute values must be *quoted*
  - Attribute minimization is *forbidden*
  - The id attribute *replaces* the name attribute
  - The XHTML DTD defines *mandatory* elements

Attribute Minimization is Forbidden

HTML	XHTML
compact	compact="compact"
checked	checked="checked"
declare	declare="declare"
readonly	readonly="readonly"
disabled	disabled="disabled"
selected	selected="selected"
defer	defer="defer"
ismap	ismap="ismap"
nohref	nohref="nohref"
noshade	noshade="noshade"
nowrap	nowrap="nowrap"
multiple	multiple="multiple"
noresize	noresize="noresize"

The id Attribute replaces the Name Attribute

- HTML 4.01 defines a `name` attribute for the elements a, applet, frame, iframe, img, and map
- In XHTML the `name` attribute is deprecated. Use id instead.
  - `<img src="picture.gif" id="picture1" />`
  - To make your XHTML compatible with today's browsers, you should add an extra space before the "/" symbol
- Both `name` and `id` attributes are designed to be used as fragment identifiers.
  - there can only be a single attribute of type `id` per element.

Mandatory XHTML Elements

- An XHTML document has three main parts:
  - A DOCTYPE declaration
  - A head
  - A body
- An xml declaration, which has three attributes, is optional but recommended:
  - <?xml version="1.0" encoding="UTF-8“ standalone=“yes”?>
  - The version attribute is required
  - The encoding attribute specifies the character encoding the document uses. The Unicode Transformation Forma (UTF) is the default in XML
  - The standalone attribute says whether a document uses an external DTD
    - A DTD is a grammar for a class of documents
- The DOCTYPE declaration is used to indicated the DTD (if there is any) that is used by an XML document
  - XML documents do not need to be valid but need to be well-formed
  - The declaration contains or points to markup declarations for the DTD grammar

A Minimum XHTML Document Template

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<title>... </title>

</head>

<body> ... </body>

</html>
```

XHTML Document Type Definitions (DTD)

- An XHTML DTD describes in precise the allowed syntax and grammar of XHTML markup.
- There are currently 3 XHTML 1.0 document types:
  - STRICT
  - TRANSITIONAL
  - FRAMESET
- These document types are distinguished in part by the degree to which they accept or do not accept deprecated HTML elements

The 3 Document Type Definitions

- XHTML 1.0 Strict
  - Use this when you want really clean markup, free of presentational clutter. Use this together with Cascading Style Sheets.
- XHTML 1.0 Transitional
  - Use this when you need to take advantage of HTML's presentational features and when you want to support browsers that don't understand Cascading Style Sheets.
- XHTML 1.0 Frameset
  - Use this when you want to use HTML Frames to partition the browser window into two or more frames.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML Concluding Notes

- The following ideas originated from HTML 4.0
  - separation of document structure from presentation
  - issues concerning accessibility and internationalization
  - the three DTD offerings (strict, transitional, and frameset)
- XHTML 1.1 (modular XHTML)
  - Small devices (like mobile devices) cannot support all XHTML functions.
  - XHTML 1.1 divides the specification into modules with limited functionality.
  - Small browsers can reduce their complexity by supporting only selected modules (but once a module has been chosen, all of its features must be supported).
  - XHTML 1.1 is a strict language. XHTML 1.1 is not backward compatible with HTML 4.
- XHTML 2.0
  - A next generation markup language.
  - The functionality is expected to remain similar to XHTML 1.1, but not intended to be backward compatible with HTML 4, XHTML 1.0 and XHTML 1.1

XHTML Validation

- An XHTML document is validated against a Document Type Definition (DTD)
- Test your XHTML with the W3C Validator
  - `http://validator.w3.org/`
- XHTML Tag List
  - `http://www.w3schools.com/xhtml/xhtml_reference.asp`
- XHTML Attributes
  - `http://www.w3schools.com/xhtml/xhtml_standardattributes.asp`

3.3

CSS (Style Sheets)

The Problem with HTML

- HTML was originally intended to describe the content of a document
- Page authors didn't have to describe the layout
  - the browser would take care of that without any formatting tag in HTML
- This is a good engineering approach, but it didn't satisfy advertisers and "artists"
- As a result, HTML acquired more and more tags to control appearance
  - Content and appearance became more intertwined
  - Different browsers displayed things differently, which is a real problem when appearance is important

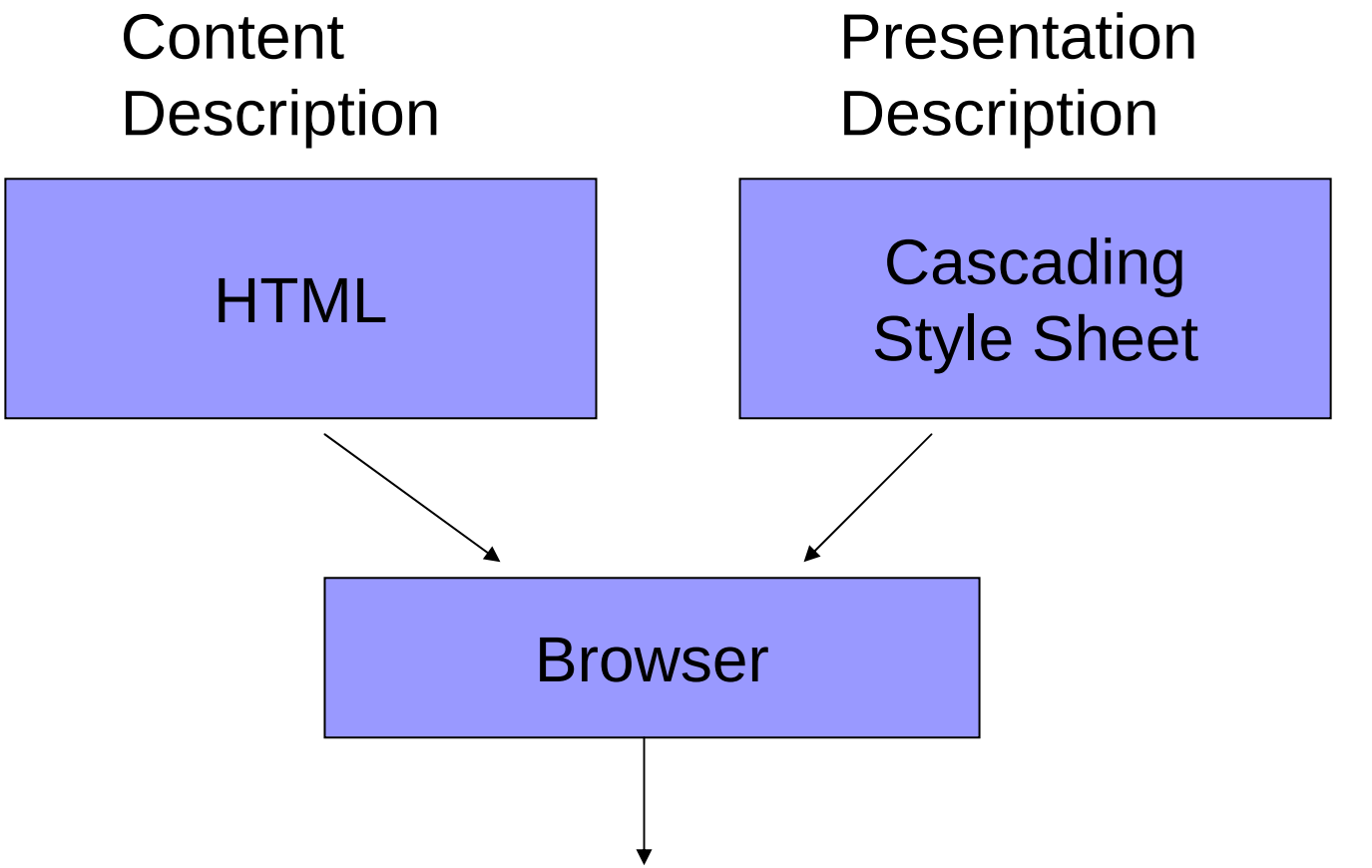
Solution: Limit HTML to...

- A language that describes what should be rendered, but not how it should be rendered
- A language that describes the hierarchy and relationships between parts of a document only

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles are normally stored in **Style Sheets**
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save you a lot of work
- External Style Sheets are stored in **CSS files**
- Multiple style definitions will **cascade** into one

Content vs. Presentation



Improved and Consistent End User Experience

CSS Observations

- It is not HTML
  - It has its own peculiar syntax
- It can be integrated into HTML
- It can be called by referencing an external file
- It can be applied globally or to a specific HTML tag

CSS Versions

- CSS 1 - Released in 1996
  - Spotty Netscape 4.x support
    - Netscape pushed their own style sheet language
  - IE 4.x was fully CSS1 compliant
  - Result: if you have users using Netscape 4.x then use CSSes with care!
    - Always test with both browsers!
  - Limitations of CSS1
    - Has almost no support for tables
    - Makes no provision for downloadable fonts
    - Lack of media types

CSS Versions

- CSS 2
  - Released in 1998
  - Extends CSS1
  - IE 5.x+ supports most, but not all CSS2 features
  - Netscape 6.x claims “unsurpassed support” for CSS1 and CSS2
  - Mozilla 1.x is generally considered to have the best CSS support
- CSS 2.1 and CSS 3 are currently under development

Compatibility Issue

- CSS1 was partially supported by browsers Internet Explorer 3, Internet Explorer 4, and Netscape Navigator 4.7
- CSS2 is fully supported by all new versions of popular Web browsers like: Internet Explorer 6, Netscape 6, Opera 5, and Micro Browsers for Mobiles
- If browser does not support CSS it will display page in HTML formatted form, ignoring the styles
  - i.e., the styles are themselves displayed

Benefits of Using CSS

- Separation of the document from the presentation
  - Easier coding and maintenance
  - Site control
- Consistency (Uniformity)
  - All pages in the site look the same
- Rich design and layout
  - Gives finer and increased control on document formatting than can be placed within HTML documents
- Accessibility
  - PC browsers, mobiles, PDAs, printers, TVs, users with disabilities, etc…
  - No browser specific requirements, such as plug-ins
- It can be used for both HTML and XML pages

Disadvantages

- The only disadvantage that can be assigned to CSS is non-compatibility with all internet browsers
- Surveys says that today 85% of users are able to see pages that use CSS, while the others are not

CSS Syntax

- The general syntax is:
  - selector {property: value}

  or
  - selector, ..., selector {
    property: value;
    . . .
    property: value
    }
  - where
    - selector is the tag to be affected (the selector is case-sensitive if and only if the document language is case-sensitive)
    - property and value describe the appearance of that tag
    - spaces after colons and semicolons are optional
    - a semicolon must be used between property:value pairs, but a semicolon after the last pair is optional
    - if the value is multiple words, put quotes around the value

… CSS Syntax

- CSS syntax is very simple -- it's just a file containing a list of selectors (to choose tags) and descriptors (to tell what to do with them):
  - Example: `h1 {color: green; font-family: Verdana}` says that everything included in h1 (HTML heading level 1) tags should be in the Verdana font and colored green
- A CSS file is just a list of these selector/descriptor pairs
  - Selectors may be simple HTML tags or XML tags, but CSS also defines some ways to combine tags
  - Descriptors are defined in CSS itself, and there is quite a long list of them

Example of CSS

```
/* This is a comment */

h1,h2,h3 {font-family: Arial, sans-serif;}          /* use 1st available font */

p, table, li, address {                             /* apply to all these tags */
  font-family: "Courier New";                       /* quote values containing spaces */
  margin-left: 15pt;                                /* specify indentation */
}

p, li, th, td {font-size: 80%;}                     /* 80% of size in containing element */

th {background-color:#FAEBD7}                       /* colors can be specified in hex */

body { background-color: #ffffff;}

h1,h2,h3,hr {color:brown;}                  /* adds to what we said before */

a:link    {color:darkred}                           /* an unvisited link */

a:visited {color:darkred}                           /* a link that has been visited */

a:active  {color:red}                               /* a link now being visited */

a:hover   {color:red}                               /* when the mouse hovers over it */
```

Types of Style Sheets

- Style sheets can be delivered to an HTML in three ways:
  - Inline (add a `style` attribute to an element):

```
<p style="font: 13pt verdana">Text</p>
```

  - Embedded (add a `style` element to specific page):

```
<head>
        <style>
                P { font: 13pt verdana; }
        </style>
</head>
```

  - Linked (add an external style definition):

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

Inline Styles

- CSS enabled browsers will recognize a new STYLE attribute for most tags
- Must use style sheet syntax for the attribute value of STYLE
- CSS attributes and values must be enclosed in double quotes
- Example:
  - `<h1 style="color:gold; font-family:sans-serif"> Applying an inline style</h1>`
- Advantage:
  - Useful if you only want a small amount of markup
- Disadvantages:
  - Mixes display information into HTML
  - Clutters up HTML code
  - Can't use full range of CSS features since contextual selectors, for example, like `li b{color:green}` may not be specifiable inline.

HTML vs. CSS Attribute Syntax

- Handling multiple attributes
  - HTML: Use one or more spaces or lines to separate attributes in the same tag
  - CSS: Separate attributes with a single semicolon (spaces and extra lines optional)
- Linking attributes with their values
  - HTML: `attribute="attribute-value"`
  - CSS: `attribute:attribute-value`

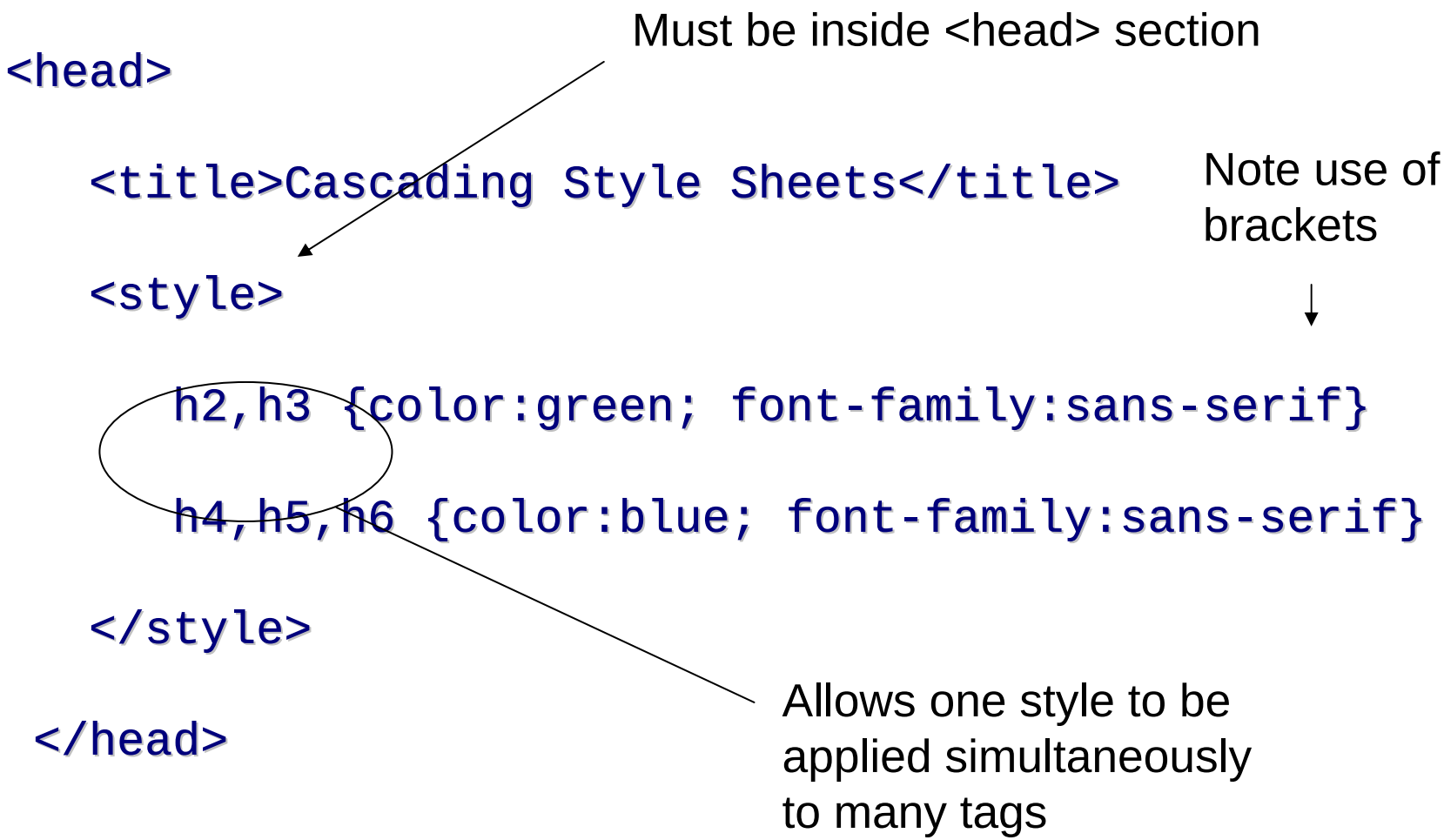
Embedded Styles

- In HTML, within the <head> element:

```
<style TYPE="text/css">
    <!--
    CSS Style Sheet
    -->
</style>
```

- Note: Embedding the style sheet within a comment is a sneaky way of hiding it from older browsers that don't understand CSS
  - These older browsers display the style sheet commands as if they are part of the document body

Embedded Style Example



External Style Sheets

- This is a file of pure CSS commands
- You apply these styles by referencing the file:
  - `<link>` tag
  - `@import` tag (offers more options)
- Both methods require a reference to the external style sheet inside the `<head>` tag
- Advantage: allows you to apply the same style easily to multiple HTML files
  - A convenient way to give a site a standard “look and feel”

Using an External Style Sheet

- Step 1: Place a number of CSS commands in a file (by convention with a `.css` extension, although `.css` is not required)
- Step 2: Reference the external style sheet with a `<link>` or `@import` command in your HTML
- Step 3: Decide if you want to selectively override the external style sheet attributes for a particular page using embedded or inline styles

CSS File

- Simply place commands in a text file using only CSS syntax in the file, ex:
  - `body {color: brown; background-color: antiquewhite}`
  - `h1 {color:brown; font-family:sans-serif}`
- Save the file with a recommended .css extension
- Must be published to a web server as any other file would be to be used on the web
- Reference in your HTML with the `<link>` or `@import` commands

CSS Reference with <link>

- `<link>` can be used to reference external files other than a CSS
- Link syntax:
  - `<link href="url" rel="relation_type" type="link_type"> … </link>`
- Link attributes
  - `href`: location of the external file
  - `rel`: must be "`stylesheet`" to tell HTML the link is for a stylesheet
  - `type`: usually "`text/css`", the MIME type needed to download the file

<link> example

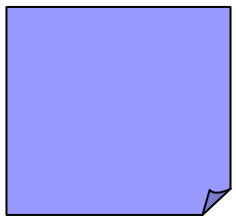
```
<head>

   <title>Cascading Style Sheets</title>

   <link href="css-2.css" rel="stylesheet"
    type="text/css" />

 </head>
```

@import

- Can be used in the `<style>` tag, or used in a .css file by itself as a CSS command
- Essentially allows for multiple inheritance of style sheets attributes
  - For example, a subsite style sheet may override a general site style sheet
  - An HTML page may override the subsite's style sheet
- Can't be used with Netscape 4.x
  - Supported by HTML 4.0 browsers only

@import Example

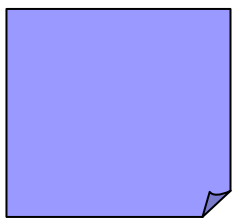
Site.css



```
h1 {color:brown; font-family:sans-serif}
… other styles ...
```

inherit

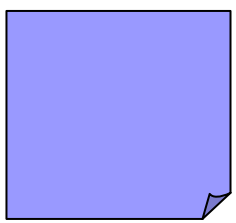
Subsite.css
(Inherits styles and overrides some styles)



```
@import url(Site.css)
h1 {color:green; font-family:Monotype}
… other styles ...
```

inherit

MyHTML.htm
(Inherits Subsite.css styles and overrides some styles)



```
<style>
   @import url(Subsite.css)
   h1 {color:red; font-family:cursive}
</style>
```

Resolving Style Preferences

- Inline style
- If no inline, embedded style is applied
- If no embedded style, external style sheet applied
- Any undefined attributes use web browser default

Cascading Order

- What style will be used when there is more than one style specified for an HTML element?
  - styles will "cascade" into a new "virtual" Style Sheet by the following rules (number four has the highest priority):
    1. Browser default
    2. External Style Sheet
    3. Internal Style Sheet
    4. Inline Style

Style Sheet Inheritance

- Tags embedded in other tags inherit style attributes

- Examples:
  - `<p>` inherits from `<body>` because it can only appear in the `<body>` section
  - `<li>` inherits from `<ol>` because `<li>` appears inside the `<ol>` tag

Style Sheet Inheritance Example

```
<body style="color:red">
    <p>This paragraph will appear with red text
      because it inherits properties of the body tag
      that encloses it.</p>
    <p style="color:green">This paragraph will appear
      with green text because it explicitly overrides
      the red text inherited from the body tag.</p>

</body>
```

Font Settings

- When text is displayed in a browser it appears in a default font face, size, style, and color.
- Most browsers use the Times New Roman font face at approximately 12-point size and rendered in black.
- CSS settings permit you to change these default settings to bring a different look to your pages.
- There are six font style settings that can be used:
  1. font-family
  2. font-size
  3. font-style
  4. font-weight
  5. font-variant
  6. font

1. `font-family` Property

- A generic description of a range of font types all having a similar design supported by all CSS capable browsers
- The `font-family` property needs to be specified to change the browser's default setting from Times New Roman.
- Five generic font families are supported by Cascading Style Sheets:
  - **Serif** (*e.g.*, Times)
  - **Sans-serif** (*e.g.*, Arial or Helvetica)
  - **Cursive** (*e.g.*, Zapf-Chancery)
  - Fantasy (*e.g.*, Western)
  - **Monospace** (*e.g.*, Courier)

Other Font Family

- A computer may provide additional font families that supplement generic font families
- You cannot assume these additional families will be available
  - So if used specify a generic font to use if the specific font family is not available
- The following font faces are typical on a Windows-based PC:
  - arial
  - arial narrow
  - comic sans ms
  - courier new
  - georgia
  - impact
  - tahoma
  - times new roman
  - verdana

Font Family Specification

- Example:
  - `h1, h2, h3, h4, h5, h6 {font-family: Arial Helvetica sans-serif}`
- As with the `<font>` tag proceed from the most unlikely to the most likely font family
  - Similar to `<font face="face">` attribute
- End with a generic font family

2. `font-size` Property

- The `font-size` property is used to change the browser's default 12-point size.
  - You can use pixels to set letter heights for special styling.
- Two ways to specify:
  - Absolute
  - Relative
    - Using a Keyword description
    - As a percent of the default font size for a tag

Absolute Font Specification

- millimeters (use `mm`)
- inches (use `in`)
- points (72 points per inch; use `pt`)
- pica (6 picas per inch; use `pc`)
- pixel (use `px`)
  - Smallest display element on computer monitor
- Can specify decimal units:
  - `h1 {font-size: 0.5in}`

Relative Font Specification

- Specify according to relationship to the standard character
- Standard characters: `em` and `ex`
- EM Unit
  - Equal to width of capital letter “M” in the default font
- EX Unit
  - Equal to the height of a lower case “x” in the default font

Why use relative units?

- Allows for scalable fonts
- Monitors vary in size of display and screen resolution
  - Specifying a relative unit ensures a uniform viewing experience across the variety of monitors rendering your page

Relative Unit Examples

- As a scalable font:
  - `body {font-size: 150%}`
- Use *descriptive keywords*: xx-small through xx-large:
  - `b {font-size: xx-large}`

3. `font-style` Property

- Specifies appearance of font
  - Browser default is the `normal` style.
- Syntax: font-style: style_type
- Style Types:
  - `normal`
  - `italic`
  - `oblique` (similar to italic)
- Example:
  - `p {font-style: italic}`

4. `font-weight` Property

- Specified the degree of “boldness” of the type
- Specified from 100-900 in intervals of 100
  - 100 is lightest
  - 900 is heaviest
- Example:
  - `p {font-weight: 300}`

5. `font-variant` Property

- Not supported by Netscape 4.x
- Attribute values:
  - normal
  - small-caps (EXAMPLE)
    - Uppercases but reduces font size
    - Specifying `normal` returns the text to standard display.

6. `font` Property

- Pools together a variety of text and font attributes
- Attribute values are positional: font-style specified first, font-variant second, font-weight last
- Example:
  - `h2 {font: italic small-caps bold}`

    instead of
  - `h2 {font-style:italic; font-variant:small-caps; font-weight:bold}`

Text Properties

- Font settings can be paired with other style sheet properties to apply additional formatting to strings of text.
- The following text properties can be paired with font settings to bring more variety to text displays.
  1. word-spacing
  2. letter-spacing
  3. line-height
  4. text-align
  5. vertical-align
  6. text-indent
  7. text-decoration
  8. text-transformation
- Word, letter and line spacing specify amount of white space to leave between words, letters and lines
- Syntax:
  - word-spacing: size
  - letter-spacing: size
  - line-height: size
- Size can be expressed as “normal” (browser determines spacing) or a specific unit

Word, Letter, and Line Spacing Examples

- `p {letter-spacing: 1 em}`
  - Might render: L e t t e r
- `p {word-spacing: 2 em}`
  - Might render: This is an example
- `p {line-height: 2}`
  - Indicates line height is twice the font size height
  - Default is 1.2

text-align

- Specifies horizontal alignment to use
  - Essentially the same as the align attribute of various HTML tags
- Syntax:
  - `text-align:left|center|right|justify`
- Example:
  - `h2 {text-align: center}`

vertical-align

- Specifies vertical alignment to use
- Syntax:
  - vertical-align:
    - `baseline|bottom|middle|sub|super|`
    - `text-bottom|text-top|top`
- Example:
  - `h2 {vertical-align: text-bottom}`

vertical-align attribute values

- `baseline`: aligns with bottom of font
- `bottom`: aligns with bottom of lowest element in the line
- `middle`: align in the middle of text
- `sub`: Subscript
- `super`: Superscript
- `text-bottom`: aligns with font's bottom
- `text-top`: aligns element with top of tallest letter
- `top`: aligns with top of tallest letter or graphic in the line

text-indent

- Used to indent first line of paragraph
- Specifying a negative number makes a hanging indent
  - Works sporadically in browsers
  - Negative indents are buggy in some browsers
- Can specify in absolute or relative units
- Example:
  - `p {text-indent: 2 em}`

text-decoration

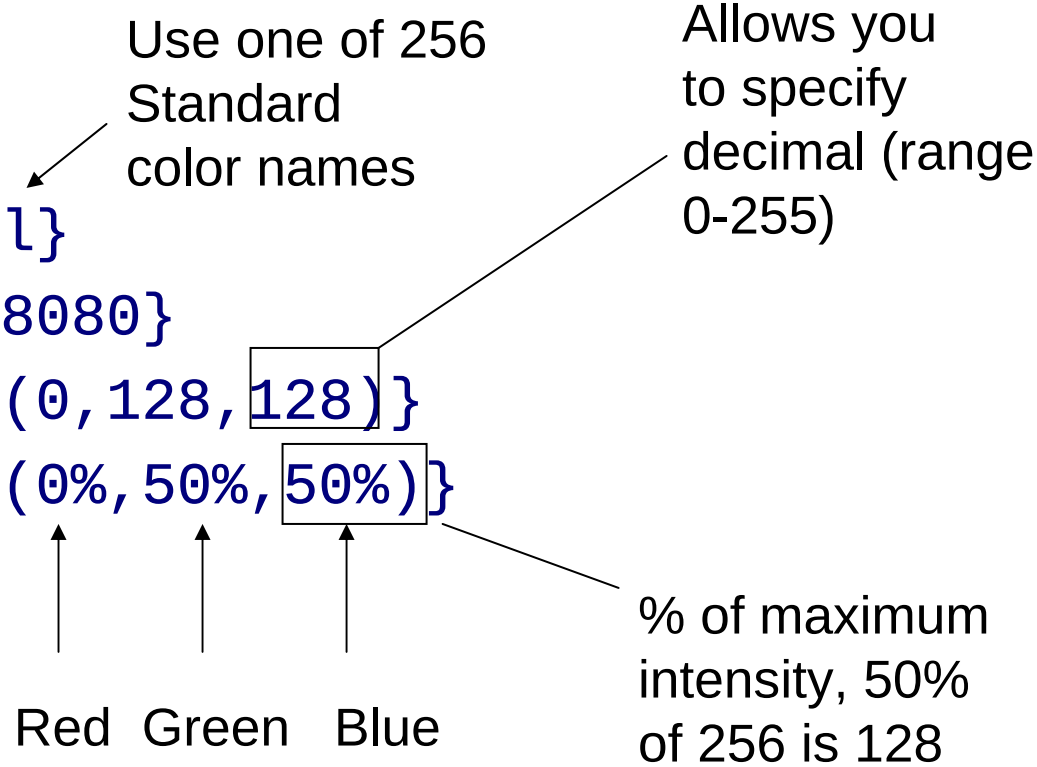
- Attribute values:
  - none
  - underline (Example)
  - overline (Example)
  - line-through (~~Example~~)

text-transform

- Attribute values:
  - none
  - capitalize (first letter of each word is capitalized)
  - uppercase
  - Lowercase
- `text-decoration` and `text-transform` affect the style of characters
  - Thus, they are better thought of as font properties

color

- Specified similar to colors in HTML
- Examples:
  - `body {color:teal}`
  - `body {color:#008080}`
  - `body {color:rgb(0,128,128)}`
  - `body {color:rgb(0%,50%,50%)}`



background-color

- Can be applied not just to body, but to almost any *block level* HTML element on web page
- Use same attributes values for color
- Example:
  - `blockquote {background-color:silver}`

background-image

- Can be applied to almost any element on a page
- If applied to an element it fills the space for that element only
- Syntax:
  - background-image:url(image)
- Example:
  - `b {background-image:url(../images/Scribble.gif)}`
  - All bold text will show this background image

background-repeat

- Controls how image is to be tiled
- Makes sense use it with a background-image
- Attribute values:
  - repeat: fill entire background
  - repeat-x: tile horizontally for width of element
  - repeat-y: tile vertically for width of element
  - no-repeat: show only once

background-position

- Specifies to offset the image by the given vertical and horizontal amount
- Makes sense to use it with background-image
- Attribute values:
  - length
  - percentage
  - top|center|bottom|right|left

background-attachment

- Makes sense to use it with background-image
- Attribute values:
  - scroll: image scrolls as text scrolls
  - fixed: keeps image in place even as text scrolls

background

- Works like font and is used to specify a variety of background attributes at once
- Syntax (attribute values are positional):
  - background:background-color background-image background-repeat background-attachment background-position
- Example:
  - `body{background: white url(squiggles.gif) no-repeat fixed center center}`

List Styles

- Expand the possibilities for how the `<ol>`, `<ul>` and `<li>` tag should be rendered

- Consists of attributes:
  - list-style-type: disc|circle|square|decimal|decimal-leading-zero|lower-roman|upper-roman|lower-alpha|upper-alpha
  - list-style-image: url(image)
    - allows an image to be used for a list item
  - list-style-position: inside|outside
    - controls whether list label is inside or outside label box

list-style

- Combines all the attributes for list styles into one attribute
- Syntax:
  - list-style: list-style-type list-style-image list-style-position
- Example:
  - `ul {list-style: circle url(Apple.jpg) outside}`

Conditional Application of CSS

- Some tags like the <a> tag allow style sheet to be applied conditionally
- Examples:
  - Visited Links (style is applied if the link has been visited)
  - Hover Links (transform test while mouse is over a link)

<a> and CSS

- In .css or in `<style>` tag:
  - `a:visited` {style definitions}

    styles to apply to visited links
  - `a:link` {style definitions}

    styles to apply to unvisited links
  - `a:active` {style definitions}

    styles to apply when link is being clicked
  - `a:hover` {style definitions}

    styles to apply when mouse hovering on link

<a> and CSS Example

- Apply in <style> tag:

```
<style>
   a:link {color: #FF0000}    /* unvisited link */
   a:visited {color: #00FF00} /* visited link */
   a:hover {color: #FF00FF}   /* mouse over link */
   a:active {color: #0000FF}  /* selected link */
</style>
```

- Styles are applied automatically as links are displayed or manipulated

Selector Types

- Type Selector
- Class Selector
- ID Selector
- Descendant Selector
- Universal Selector
- Child Selector
- Adjacent Sibling Selector
- Attribute Selector

Classes

- You can define multiple ways of formatting the same tag by multiple declaring *classes* for the tag

- Declaration example:

```
<style>
    li.sale {color:red; font-weight: 800}
</style>
```

You define the name of the class

- Usage example:

```
<li class="sale">Boots $12.99</li>
```

Apply it selectively based on context

Classes

- Classes are useful because:
  - You can specify a number of formatting styles in one definition
  - You can apply a class to numerous kinds of tags:
    - Change the definition of the class and all items are changed
- Classes can be applied to a range of tags:

```
<style>
    .NewHeader {font-style: italic}
</style>

<h1 class="NewHeader">This is in italics</h1>
<p class="NewHeader">This paragraph is in
   italics.</p>
```

ID

- ID Property works like CLASS except
  - Can only be used once in a specification
  - Cannot be applied to more than one tag
  - Example:

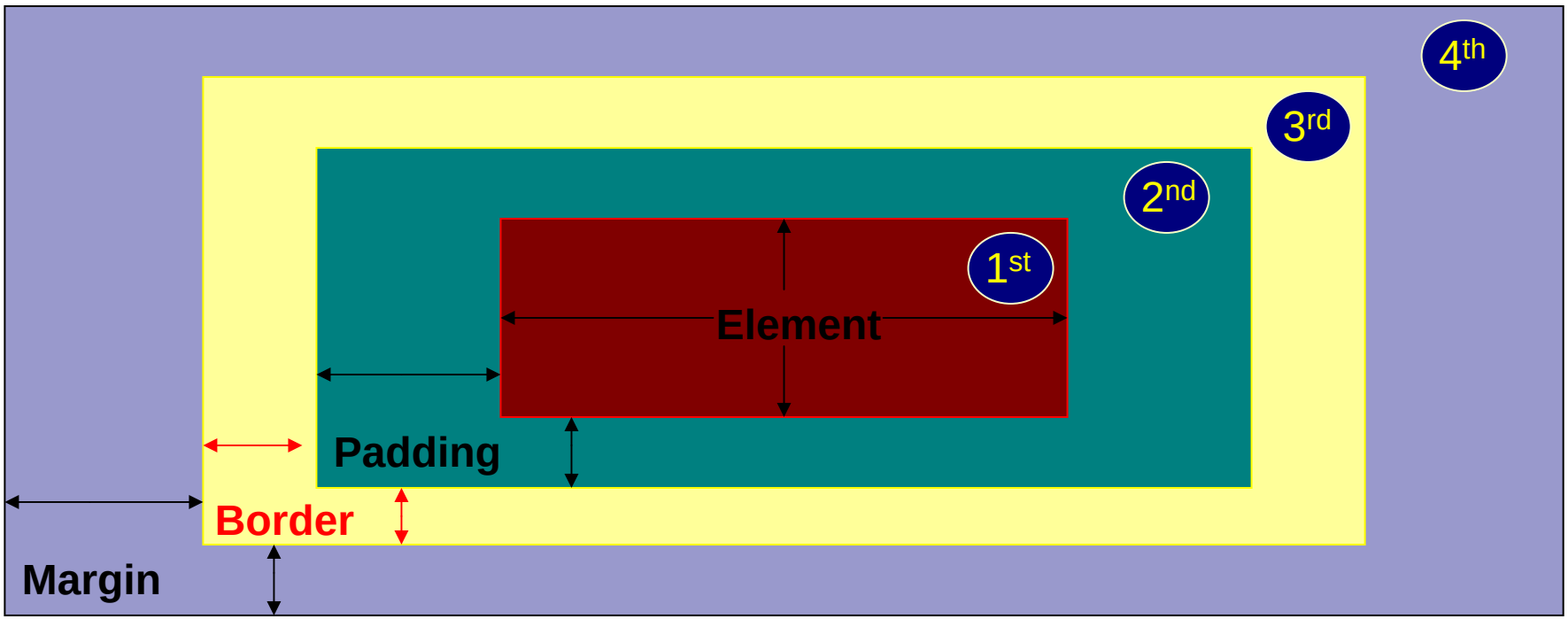
```
<style>
    #sale {color:red; font-weight: 800}
</style>
<h1 id="sale">Sale Items</h1>
<h2 id="sale">Clothing Sale Items</h2>
```

Must change id to class for this to work

Block-Level Elements

- These are HTML tags which can
  - be moved by a CSS
  - borders can be applied
  - background colors can be assigned
- They can have
  - a parent element (nested inside something else)
  - margins
  - borders
  - padding

Formatting Model



Block-Level Elements

- `<h1>` thru `<h6>`
- `<p>`
- `<blockquote>` and `<address>`
- `<ul>`, `<ol>` and `<dl>` list tags
- `<li>`
- `<div>`
- `<body>`
- `<hr>`
- `<img>`

Block-Level Elements

- Because <p> is a block level element it can use common block level attributes
  - `p {margin-left: 2em; margin-right: 2em; margin-top: 1em; margin-bottom: 1em}`

Block-Level Attributes

- Margins
  - margin-top
  - margin-right
  - margin-bottom
  - margin-left
- Padding
  - padding-top
  - padding-right
  - padding-bottom
  - padding-left

Block-Level Attributes

- Border Width
  - border-top-width
  - border-right-width
  - border-left-width
  - border-bottom-width
  - border-width
- Border Colors
  - border-top-color
  - border-right-color
  - border-left-color
  - border-bottom-color
  - border-color

Block-Level Attributes

- Border Styles
  - border-top-style
  - border-right-style
  - border-left-style
  - border-bottom-style
  - border-style
- Border values (applied to border styles)
  - solid
  - dashed
  - dotted
  - double
  - outset
  - inset
  - groove
  - ridge

Table Purpose

- “Tables should not be used purely as a means to layout document content as this may present problems when rendering to non-visual media. Additionally, when used with graphics, these tables may force users to scroll horizontally to view a table designed on a system with a larger display. To minimize these problems, authors should use style sheets to control layout rather than tables.”
  - http://www.w3.org/TR/html4/struct/tables.html#h-11.1

Alternative Design Method

- DIVs can be an alternate to <table>
- DIVs are a container like a table cell
- CSS can position the DIV

```
<div id="article">xxx</div>
#article{
   width:250px;
   padding:5px;
   float:right;}
```

DIV Design

- Use DIVs to create the skeleton of the page.
- There should be no display-specific information in the XHTML
- The Goal: separate the information from the layout / presentation of the page
- Layout is entirely controlled by CSS

DIV Design

- Identify major sections of the page
  - Masthead (Logo and Title)
  - Menu
  - Content
  - Search
  - Footer
- Don’t overuse the DIVs!
- Don’t worry about positioning in the XHTML!

tag

- `<div>` is an HTML tag
  - Does not format by itself
  - Used to logically group a sequence of *block level* tags
    - Don't try to use it to use it to group tags that are not block level, like <b>
  - Example:

```
<style>
   div.sale_items {color:red}
<style>
…
<div class="sale_items">…</div>
```

tag

- Works similar to `<div>`
- Designed to group a number of non-block level tags (inline elements) together like <b> and <i>
- Usually used to apply a style to the set of tags
- `div` and `span` are HTML elements whose only purpose is to hold CSS information
- `div` ensures there is a line break before and after (so it's like a paragraph); `span` does not

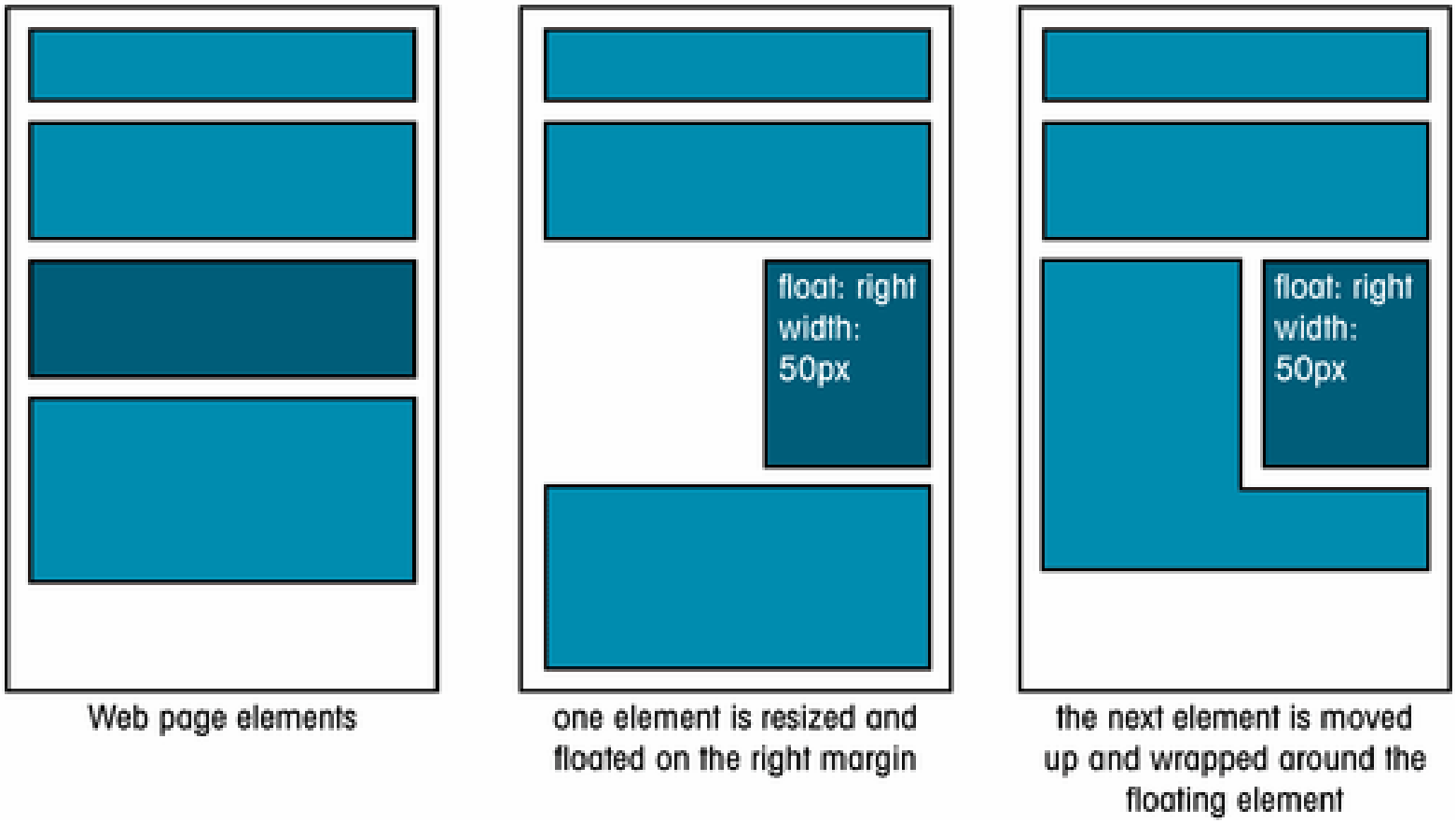
Resizing block level tags

- Any block level tag can have its width and height set
  - Be careful especially with height because if text exceeds the size of the area allocated unpredictable things might occur
- Example: Keep the <body> to 75% of the width of the browser window
  - `body {width:75%}`

Text and Block Level Elements

- Use the float attribute:
  - float:right|left
    - Text is aligned around the block level tag
  - clear:right|left|both
    - Prevents a floating element from appearing along-side a block level element

Floating a Block-Level Element

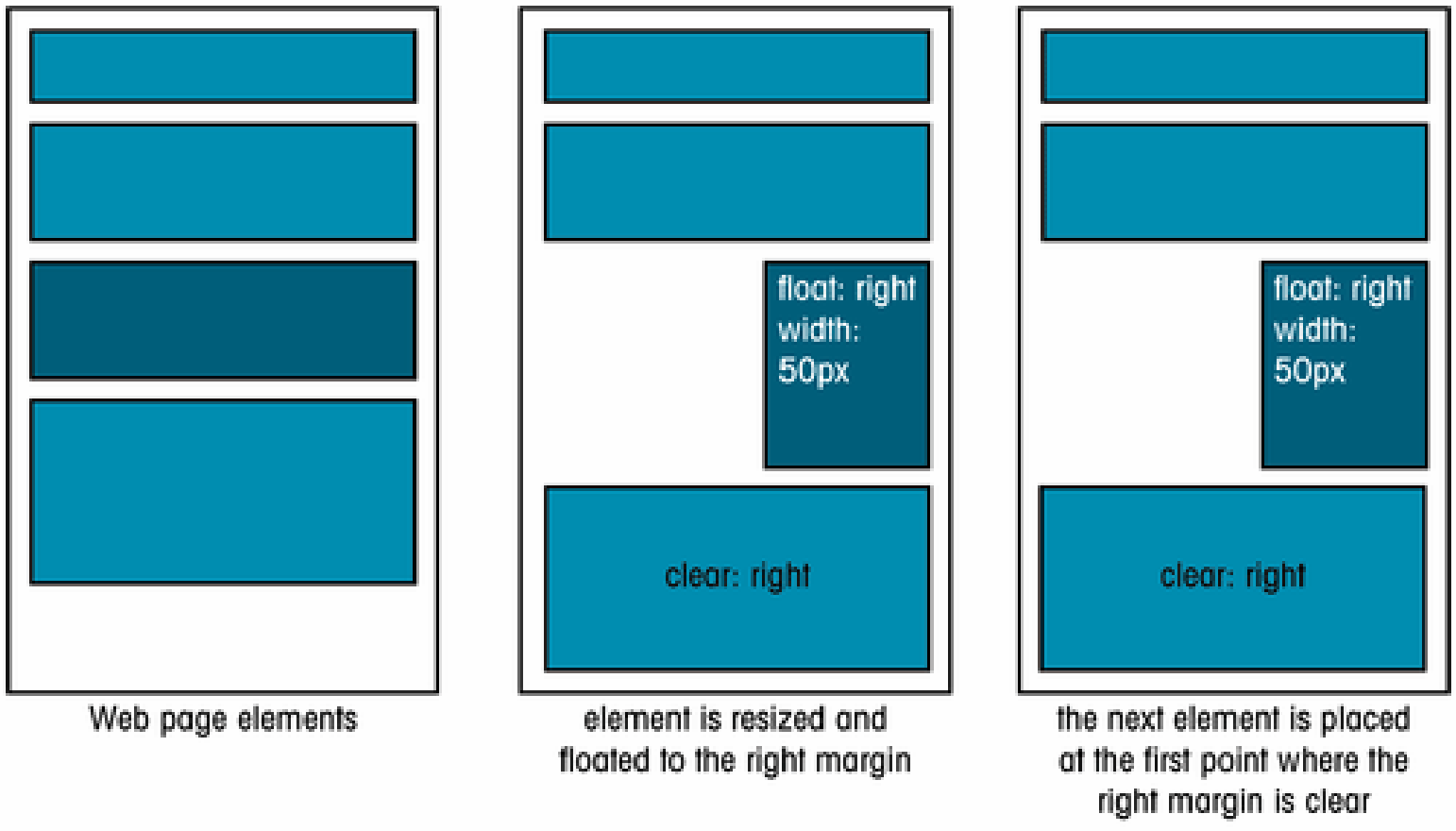


Web page elements

one element is resized and floated on the right margin

the next element is moved up and wrapped around the floating element

Using the Clear Attribute

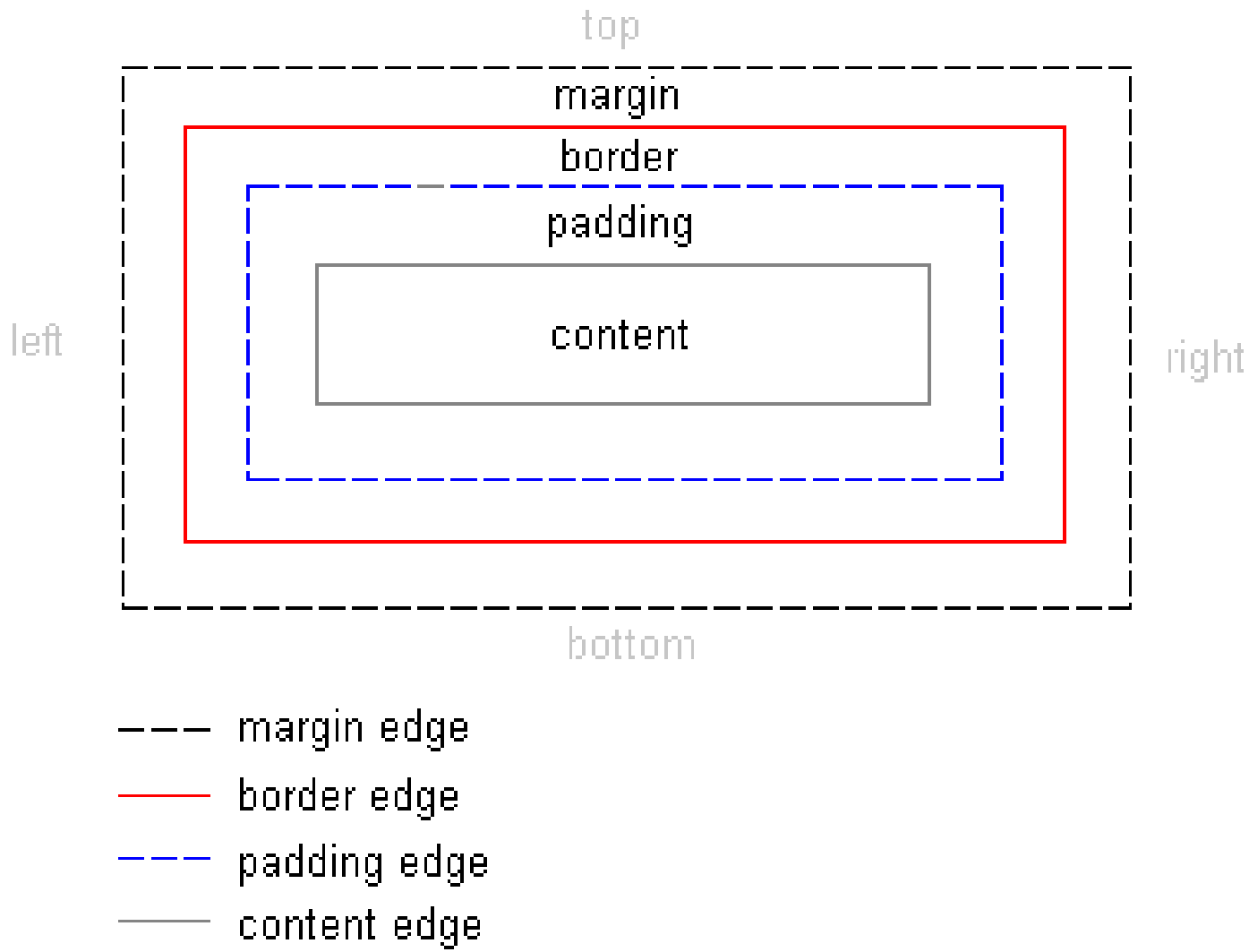


Web page elements

element is resized and floated to the right margin

the next element is placed at the first point where the right margin is clear

Tools


top
margin
border
padding
content
left
right
bottom
margin edge
border edge
padding edge
content edge

Skinning Concept

- Like a cell phone, web pages can have "face plates" (skins) that are changeable
- The CSS skins have nothing to do with the XHTML markup
- External CSS file
- Easily modifiable
- Faster Redesign

CSS Zen Garden

- The best example showing the power of CSS skinning!
- http://www.csszengarden.com

References

- W3C Cascading Style Sheets home page
  - http://www.w3.org/Style/CSS/
- CSS2 Specification
  - http://www.w3.org/TR/CSS2/
- W3 Schools CSS Tutorial
  - http://www.w3schools.com/css/default.asp
- Index DOT Css (The Advanced CSS Reference)
  - http://www.blooberry.com/indexdot/css/index.html

Resources

- CSS Editors
  - Best CSS stand alone editor is
    - Topstyle Pro – http://www.bradsoft.com
- CSS Validators
  - http://jigsaw.w3.org/css-validator/