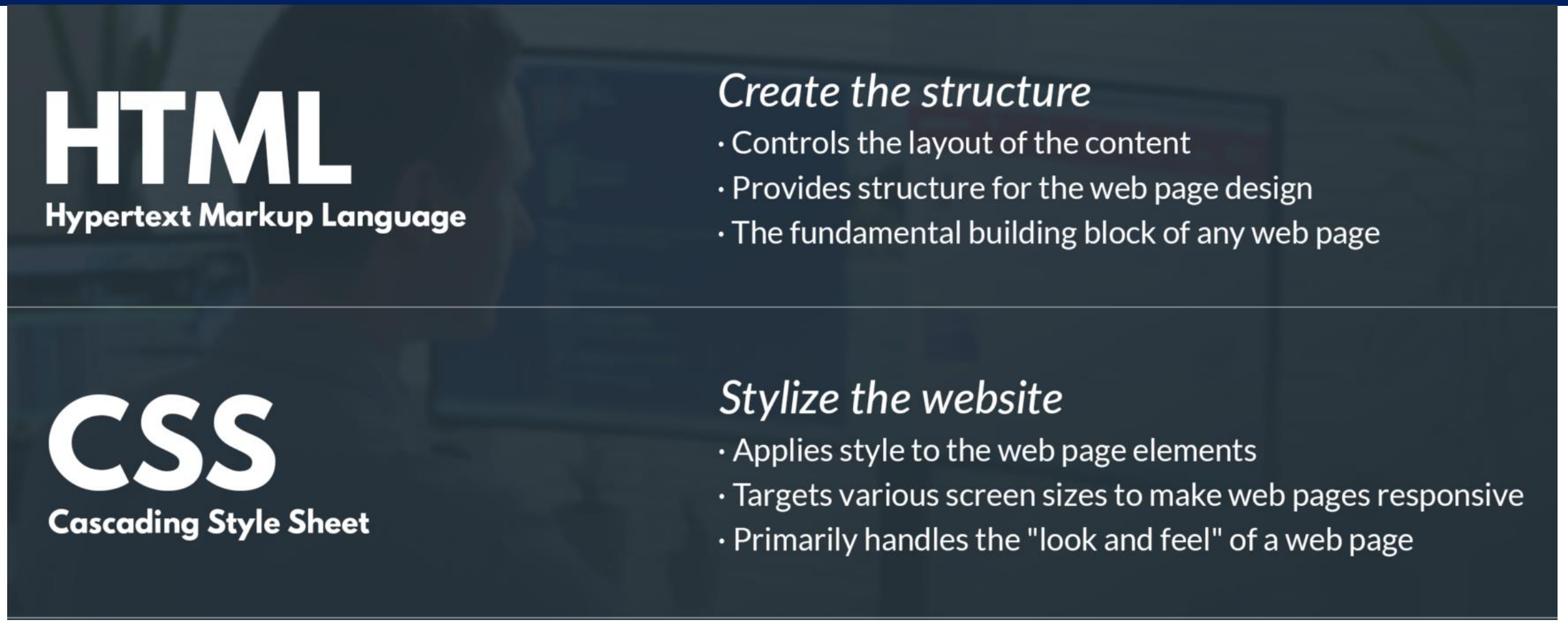
HTML AND CSS

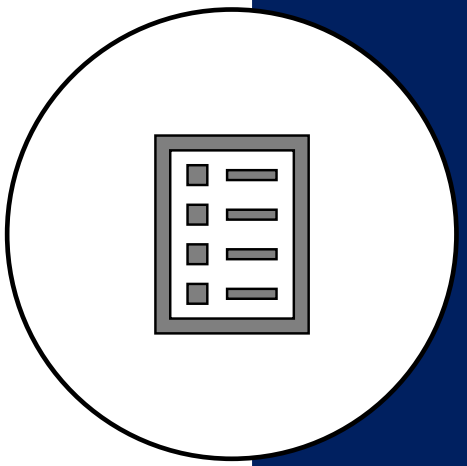
ITEC 1010: Information and Organizations



Introduction

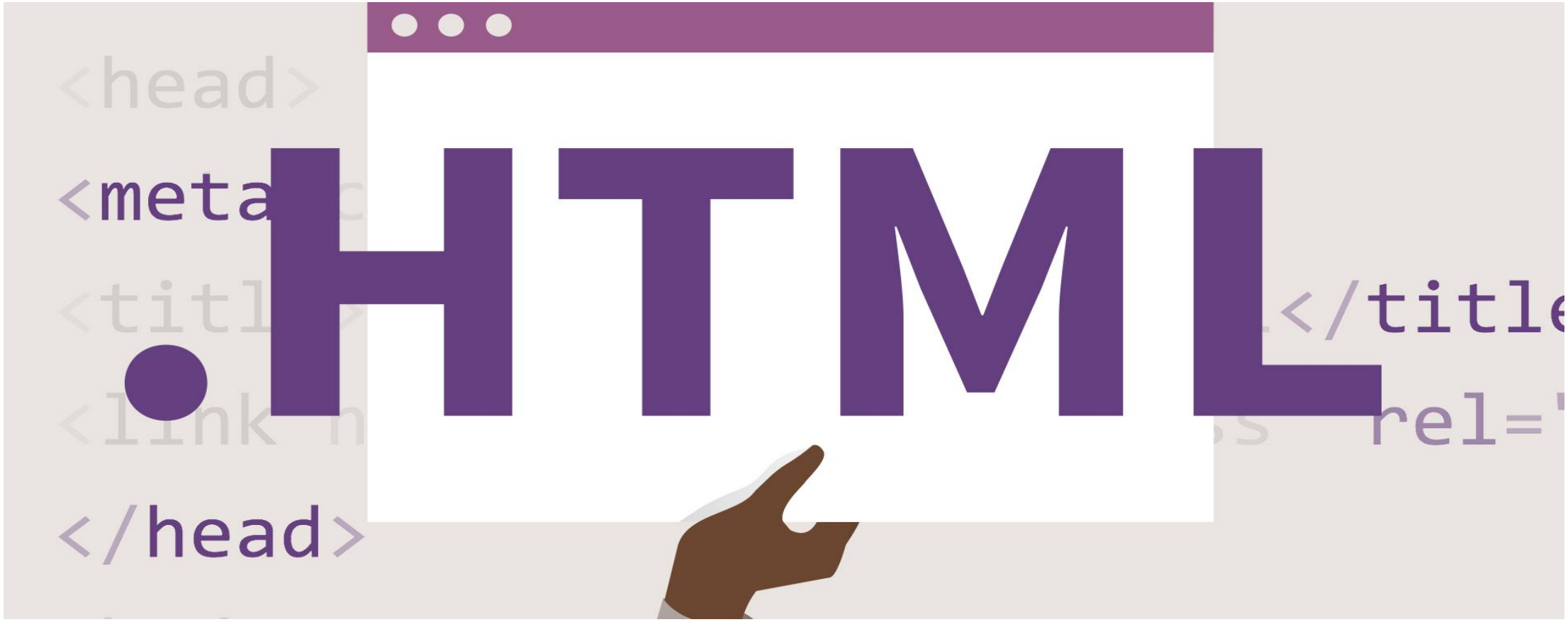
Agenda

- Introduction
  - HTML vs CSS
- HTML
  - History
  - Definition and Anatomy
  - Elements
  - Images/Links/Tables
  - Structural Tags
- CSS (Styling)
  - Intro
  - Frequently used tags
  - How to specify style
  - Relationship between rules and elements
  - Inheritance and cascading
  - Declaration values
  - Document flow
  - Backgrounds
  - Fonts
  - Text properties



HTML vs CSS

- Web pages are connected to each other using **hyperlinks**
- **Hyperlinks** are highlighted text in web page that open a new Web page with related content
- Web is a hyperlink-based system
- HTML and CSS are used to create Web pages
- **HTML used to define the structure of webpage**
- **CSS used to define the styling of a webpage**



HTML

HTML

- HTML = Hyper Text Markup Language
  - An HTML page consists of:
    - A Document Type Declaration, which triggers the rendering mode e.g., <!DOCTYPE html>
    - A system of directives on how to render content in a web browser
    - Content
- Online reference for full HTML syntax: https://www.w3schools.com/html/

HTML Template

- There are some elements that must be present in every web page → Think of these elements as a **TEMPLATE**

```
1  <!DOCTYPE html>
2  <html>
3          <head>
4          <meta charset="utf-8" />
5          <title> An HTML Page </title>
6          </head>
7          <body>
8          <! – Your web page contents follows this comment tag -->
9          </body>
10 </html>
```

HTML Template

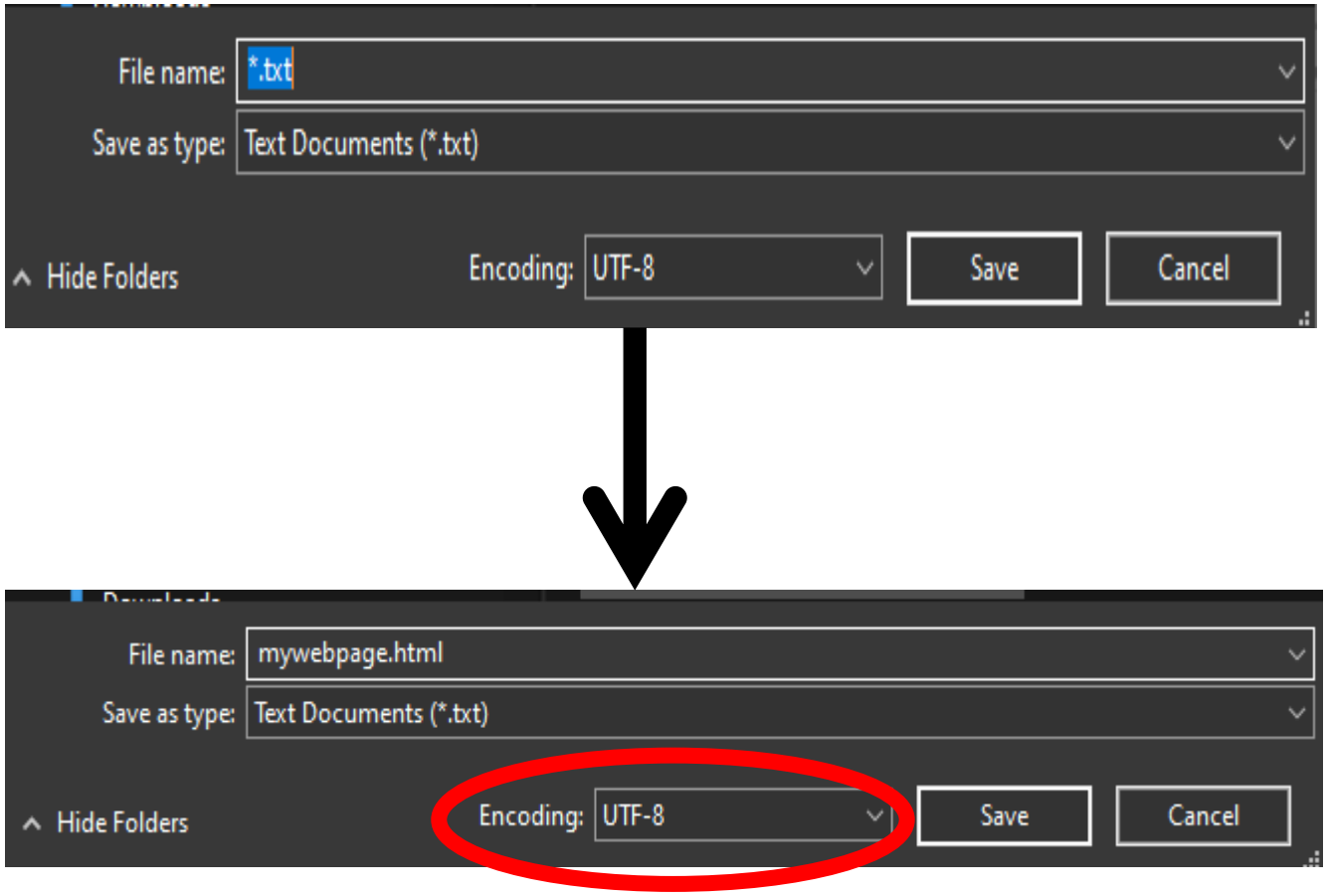
- Line 1: Identifies this document as an HTML document
- Line 2: Root level tag marking the beginning and the end of the document
  - Its end tag on line 10 is the very last tag in the document
  - Has only two child tags: *head* and *body*
- Line 3: Starts the section that tells the browser how to display the page
- Line 4: Identifies the character encoding to use
- Line 5: identifies the text to display at the top of the browser window when this page is rendered
- Line 7: Starts the section that contains your actual content
- Line 8: This is a comment. Comments are not displayed on the web page.

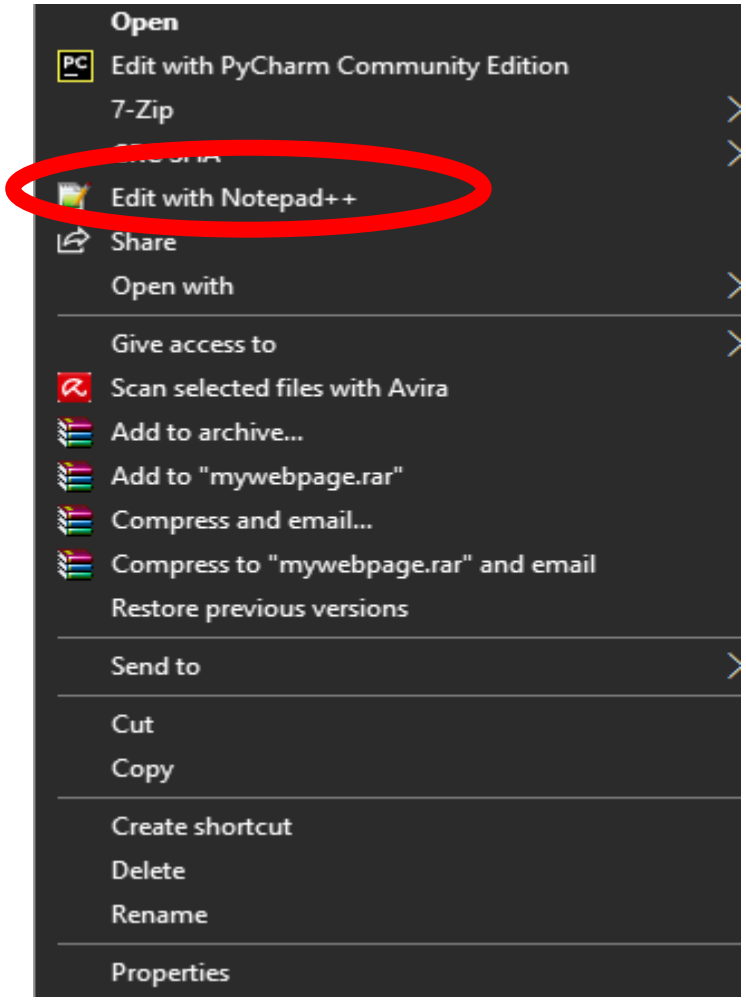
```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

How to make a web page?

- Web pages can be created and modified by professional HTML editors
- Simple text editor:
  - Notepad (PC)
  - Notepad++ (PC)
  - TextEdit (Mac)
  - Atom (Mac or PC)

How to make a web page?





HTML Elements

- HTML element is defined by start tag, content and end tag
  - Eg: <h1> My first heading </h1>
- Some HTML elements have no content and are called empty elements.
  - This is a <br> paragraph with a line break.
- There are two types of tags:
  - Enclosing tag: A pair of tags with the same name
    - Eg: for paragraph and for end of paragraph
  - Non enclosing tag: Only the start tag
    - Syntax: *<tag name* **attribute_1="value" attribute_2="value"** */>*

Frequently Used Tags

- Most of the information on a web page is text
- HTML5 provides 6 levels of Headings
  - *h1, h2, h3, h4, h5, h6* : Headings (*h1* being the highest level)
- Most text goes into paragraphs
  - *p* : Paragraph
- Lists can be ordered (numbered) or unordered (bulleted)
  - *ol* : Ordered List
  - *ul* : Ordered List
- Quotations use the *blockquote* tag
- To place text on a new line inside a paragraph, use the non enclosing Break tag *br*
- To separate text with a horizontal line, use the Horizontal Rule non enclosing tag *hr*

Nested Elements

- HTML elements can be nested so that elements contain other elements
- In nesting tags, you must place end tags in reverse order of the start tags (i.e., the end tag of the most recent start tag must come first).
  - Wrong: * * That car is ** fast * *. **
  - Right: * * That car is ** fast **. * *

Element Attributes

- All HTML elements have **attributes**
- HTML **attributes** provide the browser with additional information about the elements
  - Where to find resources
  - How to render (display) the information, including styles, colors, spacing, positioning
- Always specified in the start tag
- Attributes follow the syntax **attribute-name ="attribute-value"**
- Multiple attributes can be specified for a single tag, separated by a space
- Although sometimes certain attributes can be specified omitting the quotes, it is better to always use them

Images

- Images are tags that provide the location of a graphic file, and insert the graphic into the page
- They use the *img* non enclosing tag with:
  - The **src** attribute that specifies the location of the image file
    - The location can be:
      - A URL for images located somewhere else on the web
      - A full local specification (in Windows full path, starting with drive letter, folders and subfolders, down to actual image fine name
      - A relative specification
        - If only a file name is provided, the image file must be located in the same directory as the HTML file
        - The file name can be preceded by ./folder/ indicating that the image file is located in the named subfolder located in the same folder as the HTML file
  - The **alt** attribute that specifies what to display when the image is missing
  - E.g., *<img* **src="./Images/YorkU.jpg" alt="York University Logo"** */>*

Links

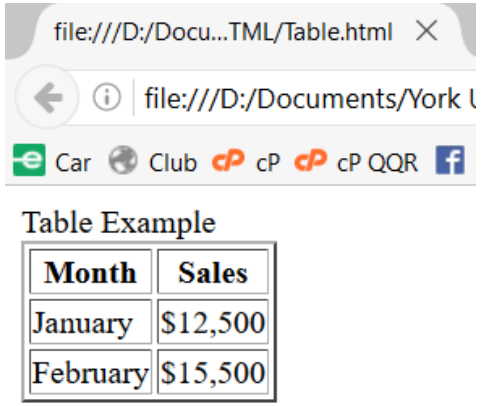
- Links are tags that provide a mechanism to navigate to another website, or another part of the same website, or to take some other action.
- They use the anchor *a* tag with:
  - The **href** attribute that specifies the target place
  - A text that will become the active link
- For navigating to another web page the **href** attribute takes a URL value
  - E.g.: *<a* **href="https://www.yorku.ca/eclass/">** York Website **
- For navigating to another spot on the same web page
  - The **href** attribute specifies a location using the syntax **"#location"**
  - The element we link to must specify the **id** attribute using the syntax **id="location"**
  - E.g.:
    - *<a* **href="#intro"**> Skip to Chapter 3 **
    - *<h1* **id="intro"**> Chapter 3 *</h1>*
- For sending an email the **href** attribute uses the syntax:
  - E.g.: *<a* **href="mailto:somebody@mailserver.com"**> E-mail Me **

Tables

- Tables are collections of data organized in rows and columns
- Column headings are placed inside Table Heading enclosing tags *th*
- The information in each cell is placed in a Table Data enclosing tag *td*
- Cells and Headings are placed inside a Table Row enclosing tag *tr*
- All table rows are placed inside a Table enclosing tag *table*
  - To show the lines of the table specify attribute **border="2"**
  - E.g.:

Table.html

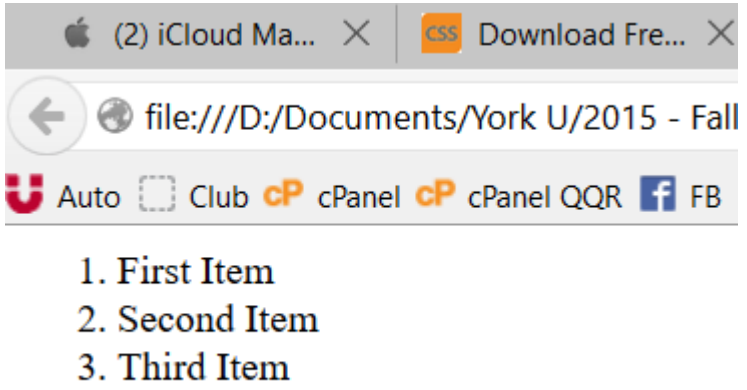
```
1  <!DOCTYPE html>
2  <html>
3      <head>
4      <meta charset="utf-8 /"
5      <title> Table Example </title>
6      </head>
7      <body>
8          <table border="2">
9              <tr> <th> Month </th> <th> Sales </th> </tr>
10             <tr> <td> January </td> <td> $12,500 </td> </tr>
11             <tr> <td> February </td> <td> $15,500 </td> </tr>
12         </table>
13     </body>
14 </html>
15
```



Lists

- Lists can be ordered (numbered) or unordered (bulleted)
  - *ol* : Ordered List
  - *ul* : Ordered List
  - The List Item tag *li* only works inside Ordered List tag *ol* or Unordered List tag *ul*

```
<ol>
        <li> First Item </li>
        <li> Second Item </li>
        <li> Third Item </li>
</ol>
```



Structural Tags – Introduction

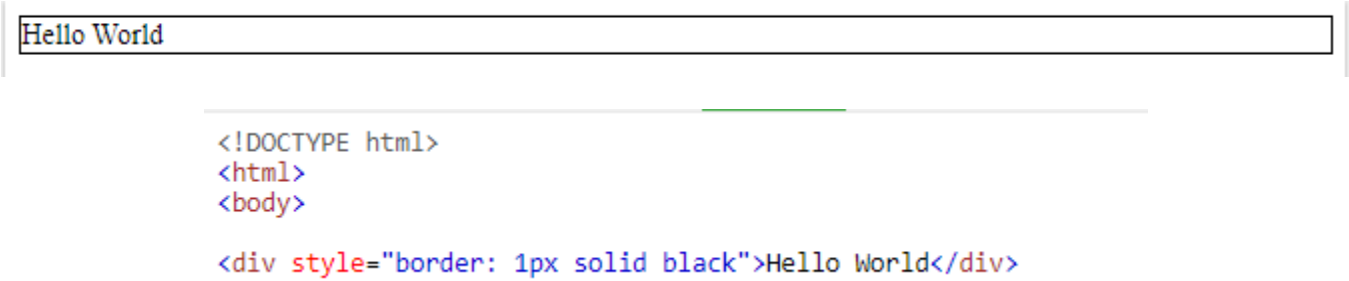
- Most tags play a dual role:
  - Be a container (box) for holding content
  - Defining the content it holds
- Other tags have no content and are simply used to group other tags into a single identifiable container. These are called **Structural Tags.**
- Most of these tags also provide a semantic meaning to the content, although not enforced
  - This helps understand the structure of the web page
- Structural tags are enclosing tags
  - We will go over div and span specifically

Structural Tags – List

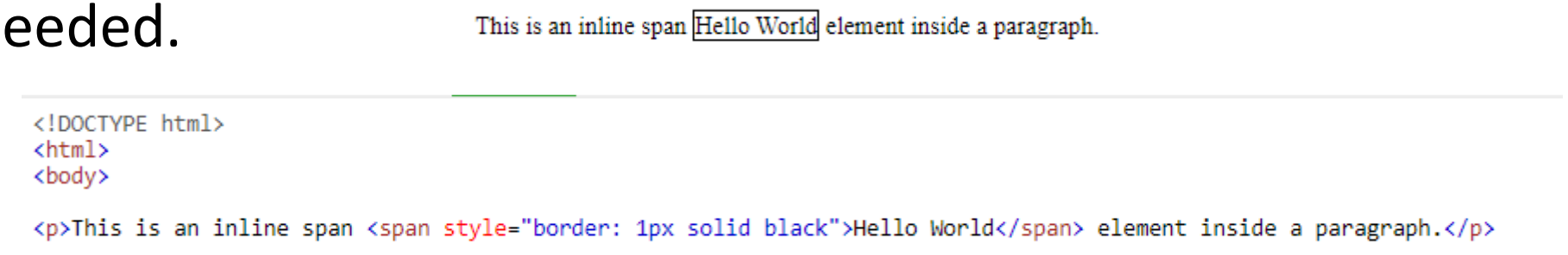
- *article* – Grouping of information on a specific topic (much like newspapers)
- *aside* – Side bar with related but non essential information (usually in an article)
- *div* – Generic high level container (stands for division)
- *figcaption* – Container for the caption text used with a figure
- *figure* – Container for graphics and illustrations related to the main text
- *footer* – Container for generic information usually located at the end of a web page
- *header* – Container for generic information (banner) located at the top of a web page
- *nav* – Container used for a collection of links used to navigate to other parts of the site
- *section* – Lower level container (usually part of a division)
- *span* – Lowest level container, typically used to group a few words for different styling

Block and Inline Elements

- Every HTML element has a default display value. There are two display values: **Block** and **Inline**
- A block-level element always takes up the full width available



- Inline elements don't start on a new line and take up as much width as needed.



Div and span

- Div
  - Block element
  - Think of div as a container that contains many other elements
  - When used in conjunction with styling, it can be used to specify style for a group of elements
- Span
  - Inline element
  - Used to mark up specific portions of text
  - When used in conjunction with styling, it can be used to specify style for a parts of content



CSS (Styling)

Styling – Introduction

- The examples of HTML we have seen are very plain looking
- HTML has many attributes that allow you to change the style of an element
- Style refers to the visual aspects of presenting the element on screen:
  - Position on screen
  - Borders, margins and spacing
  - Fonts
  - Color and size
  - … and much more…

How can style be specified?

- There are three ways to specify a CSS rule:
  - Inline: the rule is located right inside the tag it applies to, using the style attribute
    - A new paragraph
    - The scope of an inline rule is limited to that specific instance of the element
    - Note the somewhat different syntax for the rule, because it is in the form of an attribute
  - Embedded: the rule is located at the top of the *head* section, between *<style>* and *</style>* tags
    - E.g.,

```
<head>
        <style type="text/css">
            p {font-size:12px; font-weight:bold; font-style:italic; color:red;}
        </style>
</head>
```

    - The scope of an embedded rule is the web page where the style section is coded in.
  - Linked: the rule is located in a separate file (say, styles.css)
    - All pages using the file need to include a link in the *head* section, as follows:
      - <link href="styles.css" rel="stylesheet" type="text/css" />

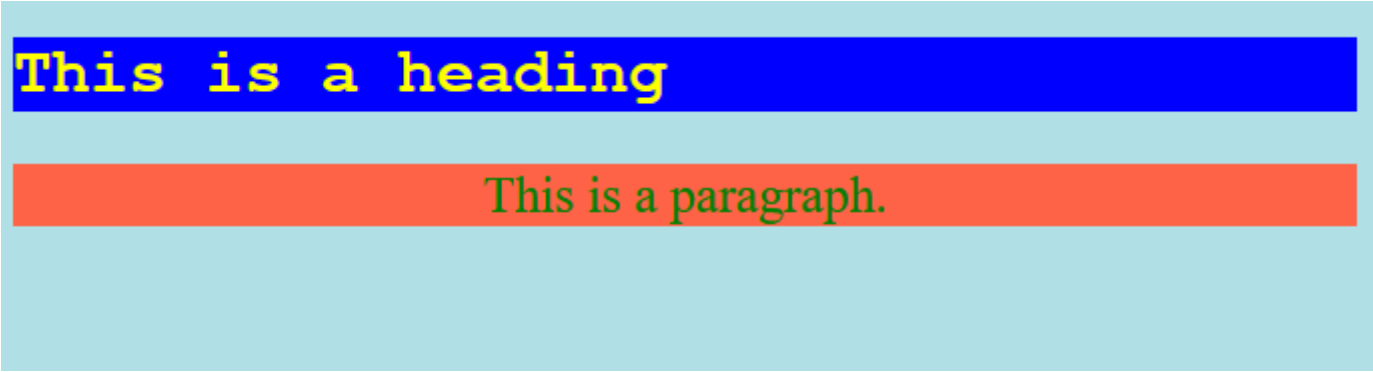
HTML Style Attribute

- The **style** attribute specifies inline style for an element
- Syntax: `<tagname style="property:value;">`

```
<body style="background-color:powderblue;">

<h1 style="background-color:blue; color:yellow; font-family:courier;">This is a heading</h1>
<p style="background-color:tomato; color:green; font-size:160%; text-align:center;">This is a paragraph.</p>

</body>
```



CSS Rules

- A CSS Rule is formed of two parts:
  - A selector (which identifies a specific HTML element)
    - There are four types of selectors
      - Tag selectors
      - Class selectors
      - ID selectors
      - Attribute selectors (we will not cover these)
  - A set of declarations (which associate styles with the selected element)
    - A declaration consists of a pair of items separated by a colon and ending with a semicolon
      - A property, identifying which aspect of style is being specified
      - A value to be attributed to the property
    - The set of declarations is enclosed in curly brackets
  - E.g., *p {color:red;}*
    - The above syntax sets up the rule that paragraphs will have their text in red, using a tag selector (*p*)

Relationship between rules and elements

- Rules and elements have a many-to-many relationship:
  - A rule may apply to many elements:
    - E.g., *h1, h2, h3, {color:blue; font-weight:bold;}*
  - Many rules can apply to the same element
    - E.g., to the rule above, you could add: *h3 {font-style:italic;}*
    - *h3* now has two rules applying to it
- Precedence: when multiple rules apply to the same element, the order of precedence is:
  - Inline rules have the highest precedence
  - Embedded rules have the second highest precedence
  - Linked rules have the least precedence
  - Think of it in terms of physical proximity to the element: the rule written closest to the element wins.

Selectors: class and id

- We can use the class attribute: **class="myclass"**
  - Identifies the specific entity it is coded on as belonging to the class **myclass**
  - You then select the this element in a CSS rule with the syntax **.myclass** (don't forget the period)
    - E.g., **.myclass** {font-style:italic;}
- Another way to identify an element is the ID attribute
  - E.g.: *<h2* **id="bio"***> Biography </h2>*
- References are identical to the syntax used for class, using # instead of . (period)
  - E.g.: **#bio** *{color:red;}*
- A common beginners mistake is to put a class attribute on all elements, and provide styling by class name
  - Defeats the purpose of providing consistent styling across multiple elements
  - If there is a true need for exception styling of one element, use an id selector

Which rule applies?

- We already learned about how physical placement of a rule affects its scope
- Still, with so many ways to create rules, it is easy to have conflicting rules
  - E.g., a paragraph with a class attribute may have one rule set up using the tag and another using the class name… which one applies?
- We have three mechanisms to determine precedence:
  - Inheritance
  - Cascading
  - Specificity (out of scope)

Inheritance

- Inheritance is based on the principle that styling applied to an element, will also be applied to its children
- Not every style element can be inherited, due to common sense:
  - Color, font, size, etc. are inherited
  - More generally, text related styles are inherited
  - Position, margins, borders, padding, etc. cannot be inherited, because it does not make sense to do so
    - Obviously, different elements need to be in different positions, or else they would overlap
    - Just because a parent box has a border, it does not follow that you want boxes inside that box to also have a border

Cascading

- We already learned that inline rules trump embedded ones, which in turn trump linked rules
- There are two more levels of this “cascading” by location:
  - Users can write their style sheets, just like a website author writes their style sheet for their website
    - Different browsers have different ways to allow users to link a stylesheet to the browser
    - Type in stylesheet and your favourite browser name in a search engine, and you will find out how to do it
  - Browsers also have a default stylesheet
    - We noticed that using different tags without any style rules still results in different styles being applied to different elements
    - These style rules come from a default browser specific stylesheet
  - Cascading precedence therefore is as follows (from lowest to highest precedence):
    1. Browser stylesheet
    2. User stylesheet
    3. Web site stylesheet
    4. Embedded styles (style section)
    5. Inline styles

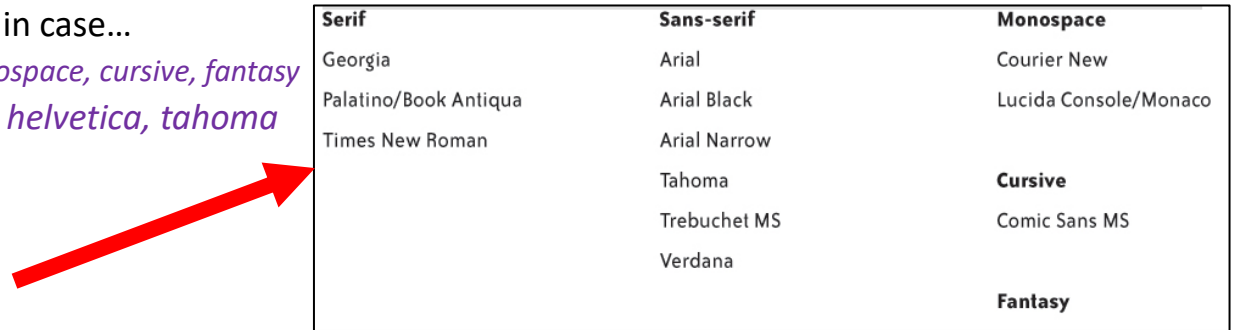


Rules of Precedence

1. Explicit rules beat implicit (inherited) rules
2. Rules located closer to the element (in the 5 level cascade hierarchy) beat rules located further away
3. If multiple rules apply at the same (closest) cascade level, the more specific rule beats the less specific one

Property: Fonts *(font-family)*

- As you recall, declarations are the second part of a rule, and consist of property – value pairs
- Most of the content of a web page is text, so typography (fonts) are very important
- There are three sources of fonts:
  - Fonts loaded on the user's computer (we will limit ourselves to these ones)
  - Fonts retrieved dynamically from dedicated font sites
  - Fonts that are downloaded with the page
    - These guarantee that a user will see the exact typography you had in mind
- Fonts are organized in:
  - Collections, containing fonts with similar graphic characteristics (serif, monospace, etc.)
  - Collections are made up of several Families (Helvetica, Times Roman, Calibry, etc.)
  - Family members are distinguished by their typeface (graphic characteristics that offer a variance on the base family type, like Bold, Oblique, Narrow, Condensed, etc.)
- ***font-family*** property specifies one or more font families in order of precedence
  - Multiple values can be specified, to cover cases where a specific font is not installed on the user's computer.
  - Font family names with two or more words must be enclosed in quotes
    - *{font-family :"trebuchet ms";}*
  - Usually, a very generic font family, is specified last, just in case…
    - There are five generic font families: *serif, sans-serif, monospace, cursive, fantasy*
  - Other commonly used font families are: *verdana, arial, helvetica, tahoma*
- There is no standard, but most computers (Mac & PC) will have these font families:



Property: Font Sizes *(font-size)*

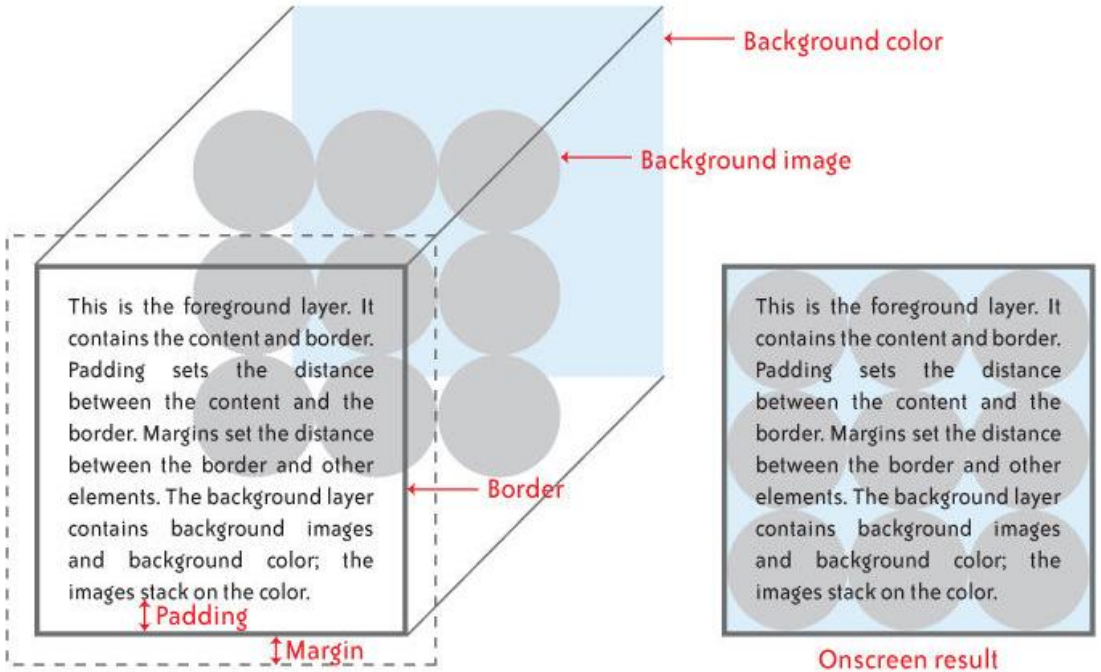
- Font sizes can be set using relative values
  - E.g., *{font-size: 50%;}* - the targeted element's fonts will be half the size of their nearest sized ancestor
  - E.g., *{font-size: 2em;}* - the targeted element's font size will be twice the size of the current font's baseline
  - The font-size property is inherited, so relative sizing has a cascading effect
- Font sizes can be set using absolute values
  - i.e., numeric sizes with units such as pixels (*px*), picas (*pc*) or inches (*in*) (*px* is the most common)
- Another size setting considered absolute is using words:
  - *x-small, small, medium, large, x-large* (medium is the same as the baseline size of the current font)
  - These are not widely used
- Font size rules
  - Default font sizes are set in the browser stylesheet in relative units
  - The font-size property is inherited
  - Therefore:
    - Any element with an absolute font size will take the size specified
    - All other elements will we sized proportionally to the ones with an absolute size
  - E.g., if you set the rule *body {font-size: 20px;}*
    - *h1* headings will be 40px (because they are sized by default to 2em, and no other rule applies)
    - *h2* headings will be 30px (because they are sized by default to 1.5em, and no other rule applies)

Other Text Properties

- While font properties style each and every individual character in text, we also need to style the appearance of entire blocks of text
- There are many text properties:
  - *text-indent*: sets the start position of the text block in relation to the containing element
  - *letter-spacing*: increases or decreases the space between letters
  - *word-spacing*: increases or decreases the space between words
  - *text-decoration*: sets visual distinctions to text (underline, overline, blink, etc)
  - *text-align*: sets the alignment style (*left, right, center,* or *justify*)
  - *line-height*: increases or decreases the space between lines of text
  - *text-transform*: changes capitalization
  - *vertical-align*: moves text up or down with respect to baseline (like subscripts or superscripts)

Property: Background

- Every element's box has two layers:
  - **Foreground**, which is taken up by the actual contents of the element
  - **Background**, which can be set to a color and an image
- Here is a tri-dimensional model of how this works:



- *{background-color:red;}* – sets the background of the box to the specified color
- *{background-image:url(photo.jpg);}* – lays the photo specified on top of the background color
  - If the photo is smaller than the box, the image is repeated left to right, then down, to fill the box, very much like tiling a wall
  - If the photo is larger than the box, the image is truncated
  - There are many ways to change this behavior, but we won't get into this

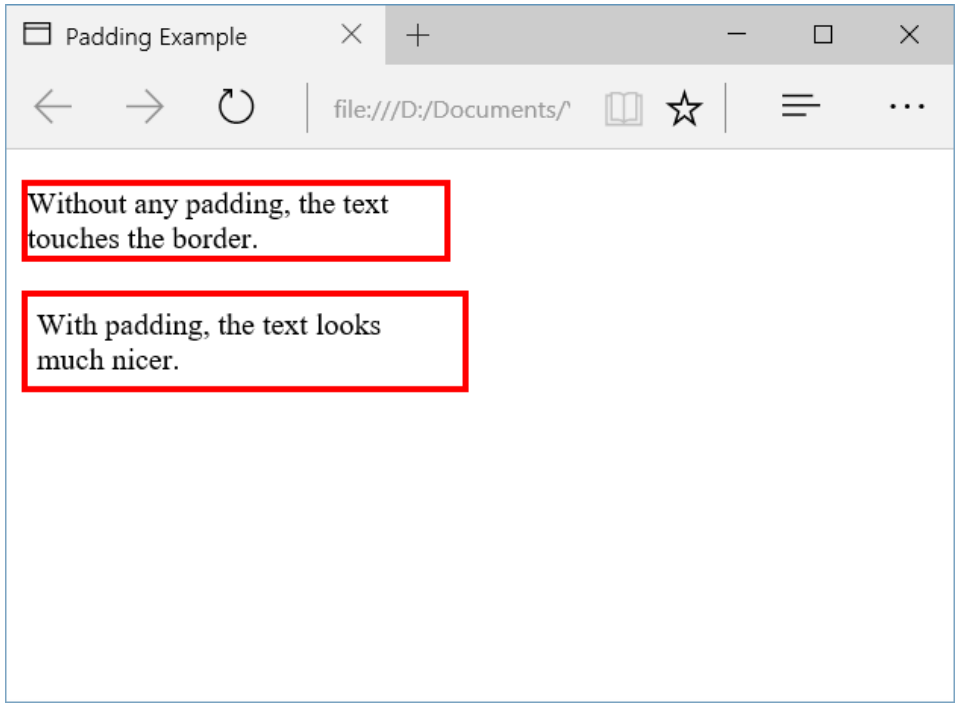
Property: Border

- The main properties of the boxes are:
  - *border*: is the "marker" of the edges of a box
    - You can set the thickness, style, and color of a border
  - *padding*: is the space between the border and the content of the box
  - *margin*: is the space outside the box, separating this box from the other adjacent elements (boxes)
- Because a box has four sides, properties associated with borders, padding, and margin have four settings: top, bottom, right, and left
  - E.g., *margin-top, border-right,* etc.
  - You can set borders, margins, and padding, separately and differently for the four sides of the box
- Borders have three properties:
  - *width:* values can be: *thin, medium, thick,* or any non-negative width unit (except %)
  - *style:* values can be: *none, hidden, dotted, dashed, solid, double, groove, ridge, inset,* or *outset*
  - *color:* values can be any legal construct for a color (see last chapter)
- To avoid having to write 12 rules for every box, we have shorthand:
  - Use the main property with one, two, or four values (order is top, right, bottom, left – i.e., clockwise from top)
    - E.g., *{border: 1px 2px 3px 4px}* – to set 4 different values for the four sides of the border
    - E.g., *{border: 1px 2px}* when a value is missing, the value for the opposite side is assumed (top and bottom 1px, right and left 2 px)
    - E.g., *{border: 4px}* – when only one value is specified, it applies to all sides
- You can specify all three properties for all four sides (most common case) like this:
  - E.g., *{border: 2px dashed red}*
- Lastly, more specific rules have priority, so you can set 3 sides of a box like this:
  - *{border: 2px dashed red}*
  - And: *{border-right: 0}*

Property: Padding

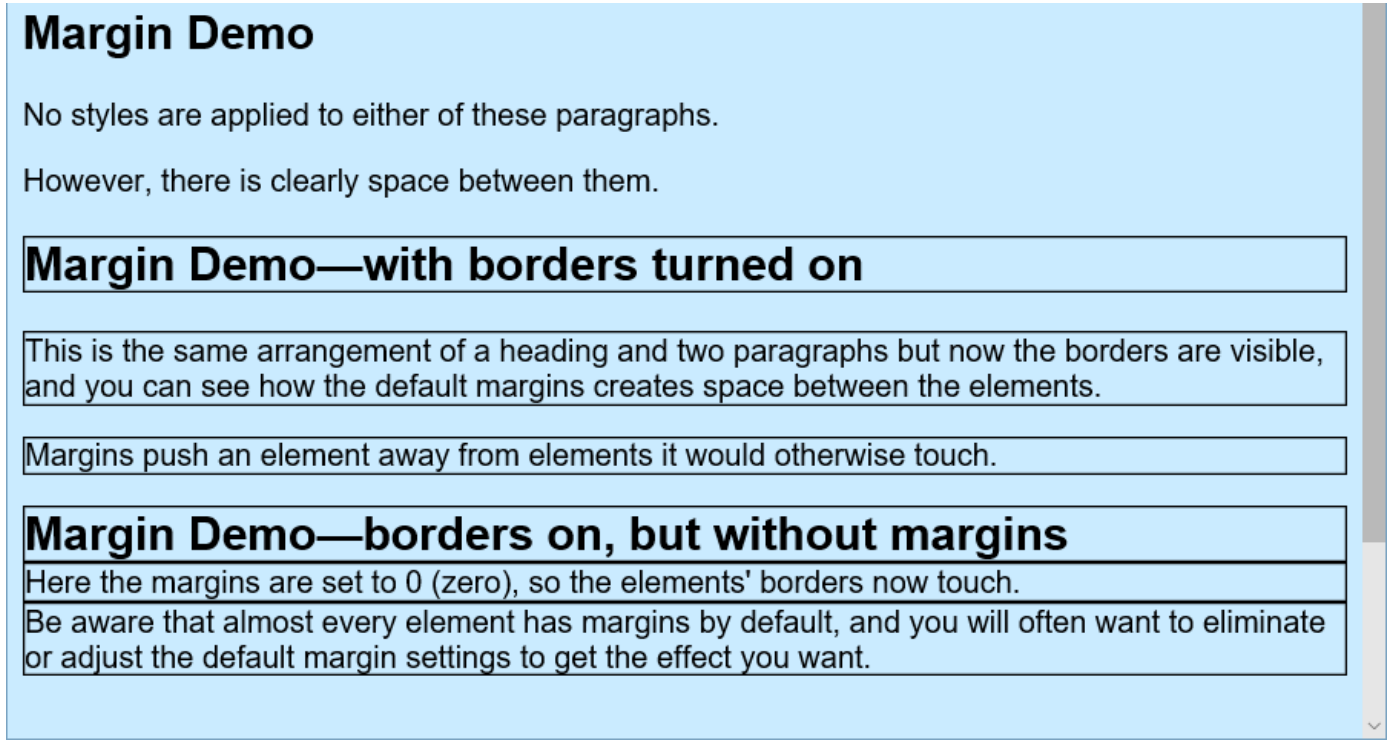
- *padding*: is the space between the border and the content of the box
- Unless you specify padding, content always touches the border (padding is set to 0 by default)
- When the border does not show, reasonable spacing does still appear, because of margins

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Padding Example</title>
    <style>
            p.nopad {border:3px solid red; width:220px;}
        p.pad {border:3px solid red; padding:5px; width:220px;}
    </style>
  </head>
  <body>
    <p class="nopad">Without any padding, the text touches the border.</p>
    <p class="pad">With padding, the text looks much nicer.</p>
  </body>
</html>
```



Property: Margin

- As opposed to padding, margins default to non-zero values
- *margin*: is the space outside the box, separating this box from the other adjacent elements (boxes)
- The easiest way to understand them is to look at the example below:



Declaration Values

- As you recall, declarations are the second part of a rule, and consist of property – value pairs
- Values can be of different types:
  - Word values (i.e., keywords) like *{font-weight:bold;}* - bold is a keyword, hence a word value
  - Numeric values are a number followed by a unit indicator, like *{font-size:12px;}*
    - Unit indicators are not required for a value of 0
  - Color values can be expressed in two formats:
    - RGB values (as in Red, Green, Blue)
    - HSL values (as in Hue, Saturation, Luminance) – out of scope
    - Sixteen standard colors have actual word values: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow.

Declaration Values: Numeric

- Absolute Values represent a numeric size as it would appear in the real world
- Relative Values size an element in relation to another:
  - em is the width of the M character in the current font, hence pixel-wise it varies based on the font
  - ex is the height of the character x in the current font, so again, pixel-wise it varies based on the font
  - % is a percentage of the current container's size, and it is very convenient for creating sizes that vary with the amount of space available

Table 2.1. Examples of Absolute Values

ABSOLUTE VALUE	UNIT ABBREVIATION	EXAMPLE*
Inches	in	height:6in
Centimeters	cm	height:40cm
Millimeters	mm	height:500mm
Points	pt	height:60pt
Picas	pc	height:90pc
Pixels	px	height:72px

Table 2.2. Examples of Relative Values

RELATIVE VALUE	UNIT ABBREVIATION	EXAMPLE*
Em	em	height:1.2em
Ex	ex	height:6ex
Percentage	%	height:120%

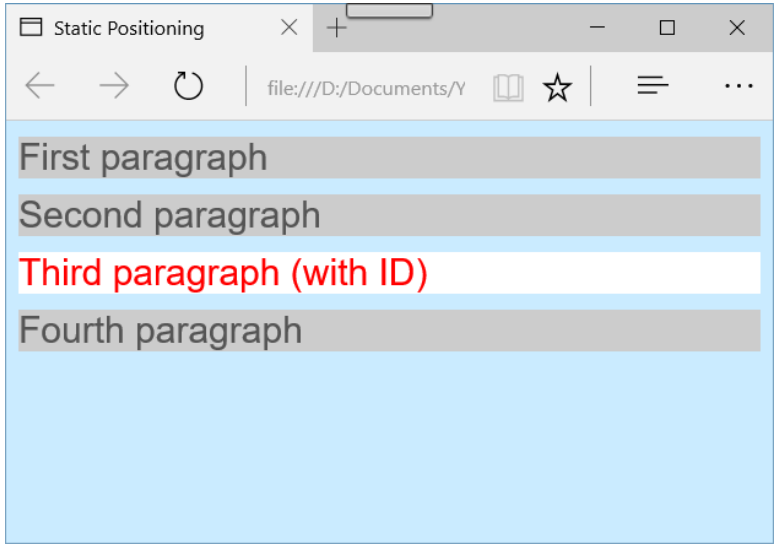
Declaration Values: Colour

- RGB color values are written in one of three formats:
  - As #rrggbb where
    - rr is a hexadecimal number between 0 and 255 representing the amount of red in the color
    - gg is a hexadecimal number between 0 and 255 representing the amount of green in the color
    - bb is a hexadecimal number between 0 and 255 representing the amount of blue in the color
      - #ff0000 is red, #00ff00 is green, #0000ff is blue
    - You can use short hand when each component has identical digits
      - #f00 for red, #0f0 for green and #00f for blue
  - As rgb(r,g,b) which is the same as above, except that r, g and b are decimal numbers
    - rgb(255,0,0) for red, rgb(0,255,0) for green, rgb(0,0,255) for blue
  - As rgb(r%,g%,b%) which is the same as above, except that r, g and b are percentages of the 0-255 range
    - rgb(100%,0,0) for red, rgb(0,100%,0) for green, rgb(0,0,100%) for blue

Document Flow: Static Positioning

- The *position* property determines where a box is displayed relative to where it would normally appear in the document flow
- It has four values: *static* (default), *relative*, *absolute*, and *fixed*
- To illustrate, we will use a four paragraph page; default positioning is where the normal document flow dictates

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Static Positioning </title>
    <style>
      p {font-size:24px; color:#555; font-family:Helvetica, Arial, sans-serif; background-color:#ccc; margin:10px 0;}
      body {background-color:#caebff;}
      p#special-para {color: red; background:#fff;}
    </style>
  </head>
  <body>
    <p>First paragraph</p>
    <p>Second paragraph</p>
    <p id="special-para">Third paragraph (with ID)</p>
    <p>Fourth paragraph</p>
  </body>
</html>
```

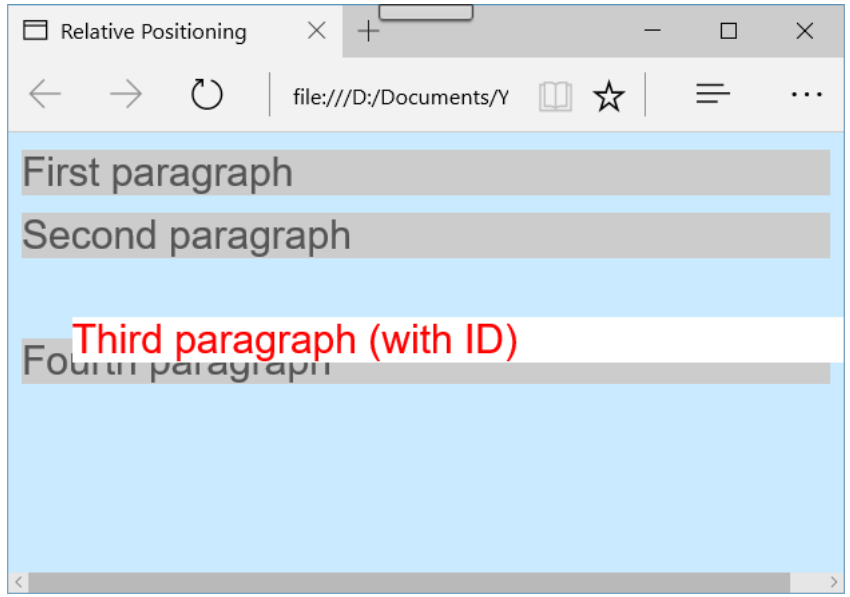


- If you change the *position* property to anything other than the default value *static*, you are positioning the element relative to another element; this other element is known as positioning context.
  - By default, the positioning context is the whole page (i.e., the *body*)

Document Flow: Relative Positioning

- Setting the *position* property to *relative* moves the box the specified amount(s), relative to where it would have gone with a default static positioning

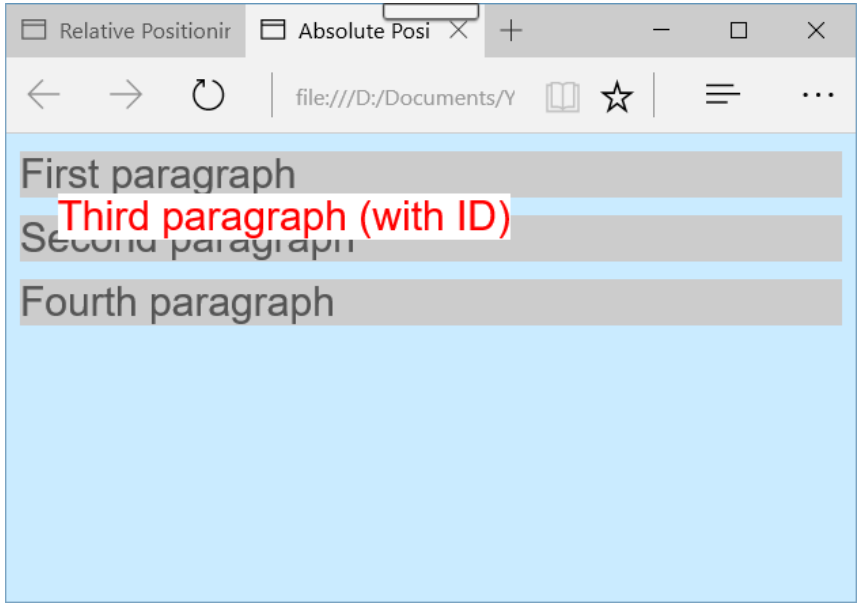
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Relative Positioning </title>
    <style>
      p {font-size:24px; color:#555; font-family:Helvetica, Arial, sans-serif; background-color:#ccc; margin:10px 0;}
      body {background-color:#caebff;}
      p#special-para {color: red; background:#fff; position:relative; top:25px; left:30px}
    </style>
  </head>
  <body>
    <p>First paragraph</p>
    <p>Second paragraph</p>
    <p id="special-para">Third paragraph (with ID)</p>
    <p>Fourth paragraph</p>
  </body>
</html>
```



Document Flow: Absolute Positioning

- Setting the *position* property to *absolute* moves the box the specified amount(s), relative to its positioning context

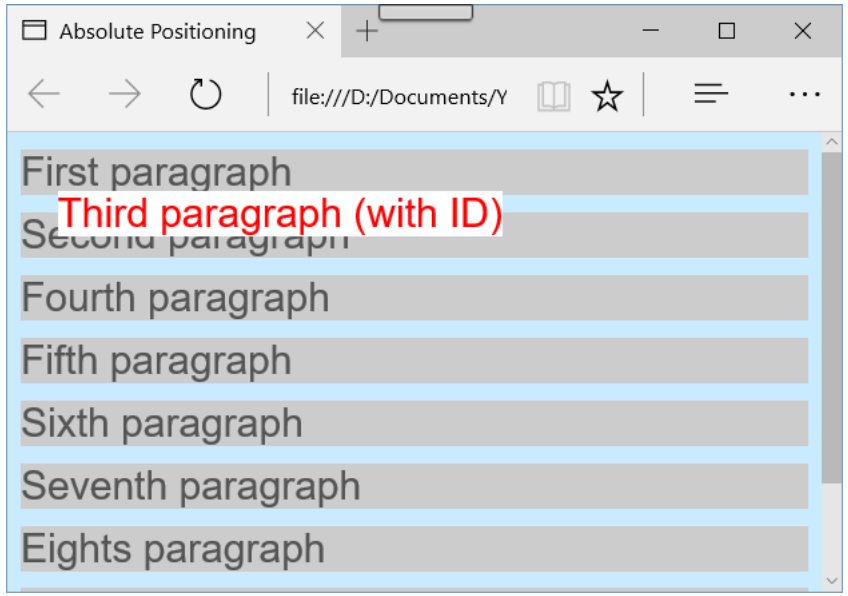
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Absolute Positioning </title>
    <style>
      p {font-size:24px; color:#555; font-family:Helvetica, Arial, sans-serif; background-color:#ccc; margin:10px 0;}
      body {background-color:#caebff;}
      p#special-para {color: red; background:#fff; position:absolute; top:25px; left:30px}
    </style>
  </head>
  <body>
    <p>First paragraph</p>
    <p>Second paragraph</p>
    <p id="special-para">Third paragraph (with ID)</p>
    <p>Fourth paragraph</p>
  </body>
</html>
```



Document Flow: Fixed Positioning

- Setting the *position* property to *fixed* is the same as setting it to absolute, except:
  - Fixed positioning is always relative to the whole page (i.e., body)
  - Fixed positioning does not scroll the element off the page

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Fixed Positioning </title>
    <style>
      p {font-size:24px; color:#555; font-family:Helvetica, Arial, sans-serif; background-color:#ccc; margin:10px 0;}
      body {background-color:#caebff;}
      p#special-para {color: red; background:#fff; position:fixed; top:25px; left:30px}
    </style>
  </head>
  <body>
    <p>First paragraph</p>
    <p>Second paragraph</p>
    <p id="special-para">Third paragraph (with ID)</p>
    <p>Fourth paragraph</p> (add a few more paragraphs like these…)
  </body>
</html>
```



Resources

- https://www.w3schools.com/html/
- https://www.tutorialspoint.com/html/index.htm
- https://html.com/

Questions?

Email: eshtab@my.yorku.ca