Faculty of Science and Technology

Department of Computer Science

WEB TECHNOLOGIES (CSC 3222)

Lecture Note 9

Week 9

Sazzad Hossain

sazzad@aiub.edu

Contents

CSS Introduction

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files**

Importance of CSS

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to **describe the content** of a web page, like:

<h1>This is a heading</h1>

This is a paragraph.

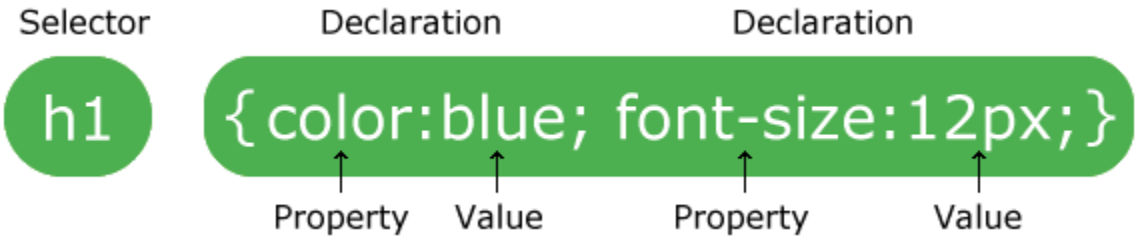
When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

Use of CSS

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example

In this example all elements will be center-aligned, with a red text color:

```
p {
  color: red;
  text-align: center;
}
```

Code	Output

<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;
  text-align: center;
}
</style>
</head>
<body>
 	
 Hello World!
 These paragraphs are styled with CSS.
 	
</body>
</html>
	Hello World!
	
	These paragraphs are styled with CSS.

- p is a **selector** in CSS (it points to the HTML element you want to style:).
- color is a property, and red is the property value
- text-align is a property, and center is the property value

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.We can divide CSS selectors into five categories:

- **Simple selectors** (select elements based on name, id, class)
- **Combinator selectors** (select elements based on a specific relationship between them)
- **Pseudo-class selectors** (select elements based on a certain state)
- **Pseudo-elements selectors** (select and style a part of an element)
- **Attribute selectors** (select elements based on an attribute or attribute value)

CSS Simple Selectors

The CSS element Selector

The element selector selects HTML elements based on the element name.

Example

Here, all <p> elements on the page will be center-aligned, with a red text color:

```
p {
  text-align: center;
  color: red;
}
```

The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example

The CSS rule below will be applied to the HTML element with id="para1":

```
#para1 {
  text-align: center;
  color: red;
}
```

Note: *An id name cannot start with a number!*

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

Example

In this example all HTML elements with class="center" will be red and center-aligned:

```
.center {
  text-align: center;
  color: red;
}
```

You can also specify that only specific HTML elements should be affected by a class.

Example

In this example only <p> elements with class="center" will be center-aligned:

```
p.center {
  text-align: center;
  color: red;
}
```

HTML elements can also refer to more than one class.

Example

In this example the <p> element will be styled according to class="center" and to class="large":

```
<p class="center large">This paragraph refers to two classes.</p>
```

The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

Example

The CSS rule below will affect every HTML element on the page:

```
* {
  text-align: center;
  color: blue;
}
```

The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

p {
  text-align: center;
  color: red;
}
```

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

Example

In this example we have grouped the selectors from the code above:

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

CSS Combinators

A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all <p> elements inside elements:

Example

```
div p {
  background-color: yellow;
}
```

Child Selector

The child selector selects all elements that are the children of a specified element.

The following example selects all <p> elements that are children of a element:

Example

```
div > p {
  background-color: yellow;
}
```

Adjacent Sibling Selector

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects all <p> elements that are placed immediately after elements:

Example

```
div + p {
  background-color: yellow;
}
```

General Sibling Selector

The general sibling selector selects all elements that are siblings of a specified element.

The following example selects all <p> elements that are siblings of elements:

Example

```
div ~ p {
  background-color: yellow;
}
```

CSS Pseudo-classes

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

```
selector:pseudo-class {
  property: value;
}
```

Anchor Pseudo-classes

Links can be displayed in different ways:

Example

```
/* unvisited link */
a:link {
  color: #FF0000;
}

/* visited link */
a:visited {
  color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}

/* selected link */
```

```
a:active {
  color: #0000FF;
}
```

`a:hover` MUST come after `a:link` and `a:visited` in the CSS definition in order to be effective! `a:active` MUST come after `a:hover` in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

When you hover over the link in the example, it will change color:

Example

```
a.highlight:hover {
  color: #ff0000;
}
```

Hover on

An example of using the `:hover` pseudo-class on a element:

Example

```
div:hover {
  background-color: blue;
}
```

Simple Tooltip Hover

Hover over a element to show a <p> element (like a tooltip):

Hover over me to show the <p> element.

Example

```
p {
  display: none;
  background-color: yellow;
  padding: 20px;
}

div:hover p {
  display: block;
}
```

CSS - The :first-child Pseudo-class

The `:first-child` pseudo-class matches a specified element that is the first child of another element.

Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

Example

```
p:first-child {
  color: blue;
}
```

CSS Attribute Selectors

The `[attribute]` selector is used to select elements with a specified attribute.

The following example selects all <a> elements with a target attribute:

Example

```
a[target] {
  background-color: yellow;
}
```

CSS [attribute="value"] Selector

The `[attribute="value"]` selector is used to select elements with a specified attribute and value.

The following example selects all <a> elements with a target="_blank" attribute:

Example

```
a[target="_blank"] {
  background-color: yellow;
}
```

CSS [attribute~="value"] Selector

The `[attribute~="value"]` selector is used to select elements with an attribute value containing a specified word.

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Example

```
[title~="flower"] {
  border: 5px solid yellow;
}
```

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

CSS [attribute|="value"] Selector

The `[attribute|="value"]` selector is used to select elements with the specified attribute starting with the specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

Example

```
[class|="top"] {
  background: yellow;
}
```

CSS [attribute^="value"] Selector

The `[attribute^="value"]` selector is used to select elements whose attribute value begins with a specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value does not have to be a whole word!

Example

```
[class^="top"] {
  background: yellow;
}
```

CSS [attribute$="value"] Selector

The `[attribute$="value"]` selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not have to be a whole word!

Example

```
[class$="test"] {
  background: yellow;
}
```

CSS [attribute*="value"] Selector

The `[attribute*="value"]` selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

Example

```
[class*="te"] {
  background: yellow;
}
```

Styling Forms

The attribute selectors can be useful for styling forms without class or ID:

Example

```
input[type="text"] {
  width: 150px;
  display: block;
  margin-bottom: 10px;
  background-color: yellow;
}

input[type="button"] {
  width: 120px;
  margin-left: 35px;
  display: block;
}
```

Inline, Internal and External Styles

There are three places where you can define style rules:

Inline Style: Included inside a particular HTML opening tag's via attribute style="style-rules". The rules are applicable to that particular HTML element only.

Internal (Embedded) Style Sheet: Embedded inside the <style>...</style> tags in the HEAD section of the HTML document. The styles are applicable to that entire document.

External Style Sheet: Stored in an external file, which is then linked to HTML documents via a <link> element in the HEAD section. The same external style sheet can be applied to all HTML pages in your website to ensure uniformity in appearance.

Inline Styles

To apply inline style to an HTML element, include the list of style properties in the style attribute of the opening tag. For example,

```
<!DOCTYPE html>
<html>
<body>
  <p style="font-size:18px; font-family:cursive">This paragraph uses 18px cursive font.</p>
  <p>This paragraph uses default font.</p>
  <p>This paragraph uses <span style="font-size:20px">20px inside this span</span>
  but default font size here.</p>
</body>
</html>
```

Output:

This paragraph uses 18px cursive font.

This paragraph uses default font.

This paragraph uses 20px inside this span but default font size here.

Take note that the `name` and `value` are separated by colon `':'`, and the `name:value` pair are separated by semicolon `';'`, as specified in the CSS syntax.

The scope of an inline style is limited to that particular element. Inline style defeats the stated goal of style sheets, which is to separate the document's content and presentation. Hence, inline style should be avoided and only be used sparsely for *touching up a document*, e.g., setting the column width of a particular table.

Internal (Embedded) Styles

Embedded styles are defined within the `<style>...</style>` tags in the `HEAD` section. For example,

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body      { background-color:cyan }
    h2        { color:white; background-color:black }
    p.cursive { font-size:18px; font-family:cursive }
    p.f20px   { font-size:20px }
  </style>
</head>
<body>
  <h2>H2 is white on black</h2>
  <p>This paragraph is normal.</p>
  <p class="cursive">This paragraph uses 18-px cursive font.</p>
  <p class="f20px">This paragraph uses 20-px font.</p>
</body>
</html>
```

- The scope of the embedded styles is the current HTML document.
- Embedded styles separate the presentation and content (in the HEAD and BODY sections) and can be used if page-to-page uniformity is not required. That is, this set of styles is used for only one single page!?

NOTE: HTML4/XHTML1 require an additional attribute type="text/css" in <style> opening tag.

External Styles

The style rules are defined in an external file, and referenced in an HTML document via the <link> element in the HEAD section.

For example, we define these style rules in a file called "TestExternal.css":

```
/* testExternal.css */
body    { background-color:cyan; color:red; }
h2      { background-color:black; color:white; text-align:center; }
p       { font-size:12pt; font-variant:small-caps; }
p.f24pt { font-style:italic; font-size:24pt; text-indent:1cm; }
#green  { color:green; }
```

This HTML document references the external style sheet via the `<link>` element in the `HEAD` section:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <link href="TestExternal.css" rel="stylesheet">
</head>
<body>
  <h2>H2 is white on black</h2>
  <h2 id="green">This H2 is green on black</h2>
  <p>The default paragraph uses 12-pt small-cap font.</p>
  <p class="f24pt">This paragraph uses 24-pt, italics font with text-indent of
1cm.
  It inherits the small-cap property from the default paragraph selector.</p>
</body>
</html>
```

You can use multiple `<link>` elements to include multiple CSS files.

The main advantage of external style sheets is that the same set of styles can be applied to all HTML pages in your website to ensure *uniformity* in presentation. External style sheet is the now-preferred approach.

NOTE: HTML4/XHTML1 require an additional attribute `type="text/css"` in `<link>` element.

Besides the HTML `<link>` element, you can also use CSS's `@import` directive to link to an external style sheet, as follows:

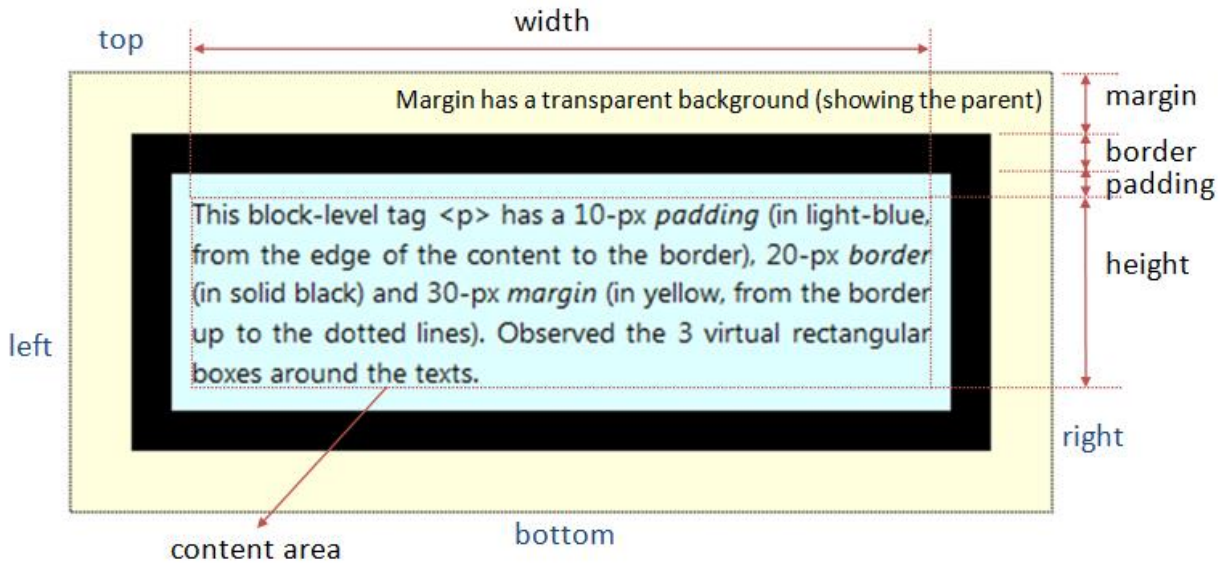
```
<style>
  @import url("cssURL1.css");
  @import url("cssURL2.css");
</style>
```

`@import` is part of the CSS language. It can also be used in one CSS file to include rules from another CSS file.

Inline styles have the highest priority, followed by internal styles, and followed by external styles.

Box Model

A block element (such as `<p>`, `<div>`, `<h1>` to `<h6>`) is always *rectangular* in shape and exhibits the so-called *box model*, with *four virtual rectangles* wrap around its "*content area*", representing the *content area*, *padding*, *border*, *margin*, as illustrated below.



1. The *content area* contains the texts, image, or child elements.
2. The *padding* is the space between the content area and the border. It clears an area outside the content area. It has the *same background* as the content area.
3. The border goes between padding and margin. You can set a color and a style (such as solid, dash, dotted) to the border.
4. The margin is the space outside the border (to another element). It clears an area outside the border. The margin does not have a background, and is totally transparent.

As illustrated in the box model diagram, margin pushes its border (and content) away with a transparent background showing the parent (having the effect of pushing itself away from the parent); while padding pushes its content inwards with the same background. Margin and padding serve the same purpose if there is no border and background applied.

Take note that the width and height that you set for the element specify its content area, exclude the margin, border and padding. To get the *actual* size of the element, you need to add the margin, border and padding to the width/height. For example, suppose that:

```
#elm {
  width: 300px;
  margin: 10px;
  border: 5px solid black;
  padding: 20px;
}
```

The *actual width* of the element is `300+(10+5+20)x2 = 370px`.

Each of the rectangular bounds has four sides, and can be individually referred to as *xxx*`-top`, *xxx*`-right`, *xxx*`-bottom`, and *xxx*`-left` in a clockwise manner, where xxx could be `margin`, `border` or `padding`. The four sides can be controlled individually or as a group.

CSS Margin, Border and Padding Properties

The margin, border and padding related properties are:

- `margin-top: auto|n|n%`
  `margin-right: auto|n|n%`
  `margin-bottom:auto|n|n%`
  `margin-left: auto|n|n%`
  Set the four margins individually. The "n" shall be expressed in a proper unit (e.g. 10px and 1.2em). You could use a negative value to overlap two elements (e.g., margin-left:-100px). The value of "auto" lets the browser to compute an appropriate number. "n%" is relative to the same property (i.e. margin-xxx) of the parent.
- `margin: margin-top margin-right margin-bottom margin-left`
  `margin: margin-top-bottom margin-right-left`
  `margin: all-4-margins`
  These are one-line shorthand notations to set all the four margins. If four values are given, they are applied to top, right, bottom, left (in the clockwise manner). If two values are given, they are applied to top-and-bottom, left-and-right. If one value is given, it is applied to all the four borders. Take note that there is no commas between the items, as all items are considered to be one property value.
- `padding-top: n|n%`
  `padding-right: n|n%`
  `padding-bottom: n|n%`
  `padding-left: n|n%`
  Set the four paddings individually, similar to margin-xxx.
- `padding: padding-top padding-right padding-bottom padding-left`
  `padding: padding-top-bottom padding-left-right`
  `padding: all-4-padding`
  A one-line shorthand notation to set all the four paddings, similar to margin.
- `border-width: thin|medium|thick|n`
  Set the width of the four borders. "n" can be used to set an absolute thickness. border-width is a shorthand notation, you can use border-width-top, border-width-right, border-width-bottom, and border-width-right to set the four borders individually.
- `border-style:`
  `none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset`
  Set the style of the 4 borders. Similarly, you can use border-style-top, border-style-right, border-style-bottom, and border-style-right to set the four borders individually.
- `border-color: #rrggbb|rgb(r,g,b)|rgba(r,g,b,a)|color-name`
  Set the color of the 4 borders. Similarly, you can use `border-color-top`, `border-color-right`, `border-color-bottom`, `border-color-right` to set the four borders individually..
- `border: border-width border-style border-color`
  Shorthand notation to set all the properties of the borders, in the order shown. You can also use the `border-top`, `border-right`, `border-bottom`, and `border-left` to set the four borders individually.

Margin, border, padding, width are NOT inherited by its descendants.

CSS Position

Each element has a natural location inside a page's flow, in the order read in by the browser. The property `position` can be used to alter the position of block elements.

`position: static|absolute|relative|fixed`

By default, elements are displayed from top to bottom in the normal flow. For block elements, line breaks are inserted at the beginning and the end to form a rectangular box. You can leave the box in the default normal flow (`position:static`); you can remove the box from the normal flow and specify its location with respect to either its parent element (`position:absolute`) or the browser window (`position:fixed`); you can also move the box with respect to its normal position in the flow (`position:relative`).

For non-static positioned elements, the new position is specified via `top`, `left`, `bottom`, `right`, `width`, `height` properties:

- `top: n|n%|auto`
  `left: n|n%|auto`
  `bottom: n|n%|auto`
  `right: n|n%|auto`
  Set the distance from the edge of this element to the corresponding edge of the containing block.
- `width: n|n%|auto`
  `height: n|n%|auto`
  Set the width and height of this block. You can use the `width` and `height` to scale this block.
- `z-index: number|auto`
  When two blocks overlap due to re-positioning, the one with larger z-index number is on top (i.e., z-axis is pointing out of the screen as in the standard 3D graphics coordinates system). Negative number is allowed. The default auto stacks the element at the same level as its parent. If the z-index of two elements are the same or no z-index are defined, the last element rendered is placed on top. z-index with alpha can create see-thru effect.

`overflow: auto|hidden|scroll|visible|inherit`

Specify how to handle content overflowing the block's width/height.

Relative Positioning "position:static"

The default `position:static` positions the element according to the normal flow of the page, in the order that is read by the browser. Properties `top`, `right`, `bottom`, `left` has no effect for `static`.

Relative Positioning "position:relative"

Move the element *relative to its normal location*. The original space occupied by this element is preserved. The surrounding elements are not affected. For example,

```
div#up {
  position: relative;
  top: -2em;  /* move this element up by 2em */
}
```

Absolute Positioning "position:absolute"

Position the element relative to the first non-static ancestor element; or <body> if no such element is found. Absolute-positioned element is taken out from the normal flow, as if it does not present.

To absolutely position an element in a containing element (other than <body>), declare the containing element relative without any movement, e.g., container { position:relative }.

Absolute positioning is interesting. You can create animation (such as bouncing ball and falling snow) by absolutely position (and repeatedly re-position) images on the browser's screen. See my "JavaScript Examples".

For example,

```
#left-panel {
  position: absolute;
  left: 10px;     /* from left edge */
  top: 10px;      /* from top edge  */
  width: 200px;
  height: 600px;
  background: black;
  color: white;
}
#main-panel {
  position: absolute;
  left: 220px;    /* from left edge 10+200+10 */
  top: 10px;      /* from top edge, same as left-panel */
  width: 560px;
  height: 600px;
  background: cyan;
  color: black;
}
<body>

<div id="left-panel">
  <h2>Left Panel</h2>
  <p>This paragraph is on the left panel</p>
</div>
<div id="main-panel">
  <h2>Main Panel</h2>
  <p>This paragraph is on the main panel</p>
</div>
</body>
```

Fixed Positioning "position:fixed"

The element is fixed at the position relative to the browser's window, and it does not scroll away. The position is defined in top, left, bottom, right (or width and height) properties. Fixed-positioned element is taken out of the normal flow, as if it is not present.

For example, a fixed is added to the above example in absolute positioning. Take note that z-index is used to ensure that the fixed is always on top of the other 's, regardless of the order of writing the 's.

```
#left-panel {
```

```
  position: absolute;
  left: 10px;     /* from left edge */
  top: 10px;      /* from top edge  */
  width: 200px;
  height: 600px;
  background: black;
  color: white;
  z-index: 100;
}
#main-panel {
  position: absolute;
  left: 220px;    /* from left edge 10+200+10 */
  top: 10px;      /* from top edge, same as left-panel */
  width: 560px;
  height: 600px;
  background: cyan;
  color: black;
  z-index: 100;
}
#fixed-panel {
  position: fixed;
  left: 100px;
  top: 200px;
  width: 200px;

  height: 200px;
  background: lime;
  color: red;
  z-index: 200;   /* larger z-index is on top */
}
<body>

<div id="left-panel">

  <h2>Left Panel</h2>

  <p>This paragraph is on the left panel</p>

</div>

<div id="main-panel">

  <h2>Main Panel</h2>

  <p>This paragraph is on the main panel</p>

</div>

<div id="fixed-panel">
  <h2>Fixed Panel</h2>
  <p>This panel does not move when your scroll up/down.</p>
</div>
</body>
```

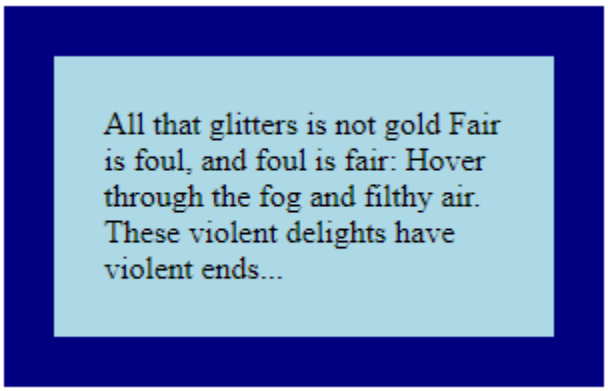
Exercises

1. Write a program to display use of element, id, class and group selectors.
2. Complete the code

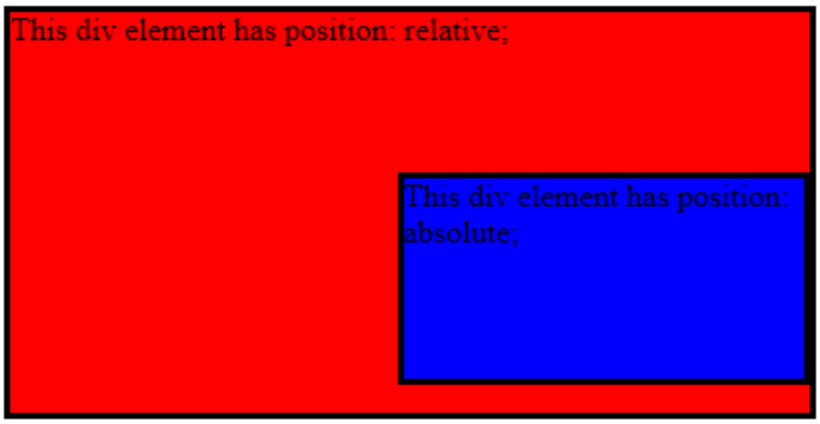
```
<!DOCTYPE html>
<html>
<head>
<style>
div {

 /*Write the code */


}
</style>
</head>
<body>

<div>All that glitters is not gold
Fair is foul, and foul is fair: Hover through the fog and filthy air.
These violent delights have violent ends...</div>

</body>
</html>
```

- Set the margin of the div to "25px".
- Set the border of the div to "25px solid navy".
- Set the padding of the div to "25px".
- Set the width of the div to "200px".
- Set the background-color to lightblue.

The output should look like below



3. Write a program to display below



Use CSS position relative and absolute to complete the code.

References

- www.w3schools.com
- https://www.ntu.edu.sg/home/ehchua/programming/