Temasek Junior College

2023 JC2 H2 Computing

Web Applications 2 - CSS Basics

Objectives

- Explain how CSS only describes the presentation of a web page and not its structure or contents.
- Use the **`<link>`** structural tag to associate a web page to a CSS style sheet.
- Identify the rules, selectors, declarations and properties that are present in a given CSS style sheet.
- Use an element selector to limit effect of a CSS rule to a particular element type.
- Use an **`id`** attribute in HTML and an ID selector in CSS to limit effect of a CSS rule to a single element.
- State that it is an error for a web page to have more than one HTML element with the same **`id`** attribute.
- Use a **`class`** attribute in HTML and a class selector in CSS to limit effect of a CSS rule to elements with the specified class.
- Use a descendent selector in CSS to limit the effect of a CSS rule to elements that are nested in a particular order.
- Distinguish between elements that have a block appearance by default and elements that have an inline appearance by default.
- Identify and distinguish between the margin, border and padding of elements that have a block appearance.
- Use the **`<div>`** tag to represent structural elements that should have a block appearance by default and are not represented by other HTML tags.
- Use the **`<span>`** tag to represent structural elements that should have an inline appearance by default and are not represented by other HTML tags.
- Use the common, box-model and typography CSS properties to control the presentation of a web page without producing any syntax errors.

1 Why CSS?

CSS stands for **Cascading Style Sheets**. With CSS, the appearance of plain webpages can be improved greatly with minimum change to the HTML script.

The table below gives a quick comparison between a webpage with and without CSS.

Without CSS	With CSS
Background is white by default	Background color can be customized
Text is black by default	Text color can be customized
Text uses a serif font by default e.g. Times New Roman	Text font can be customized
Tables are displayed without borders unless the border attribute is defined	Tables can be displayed with customized borders.

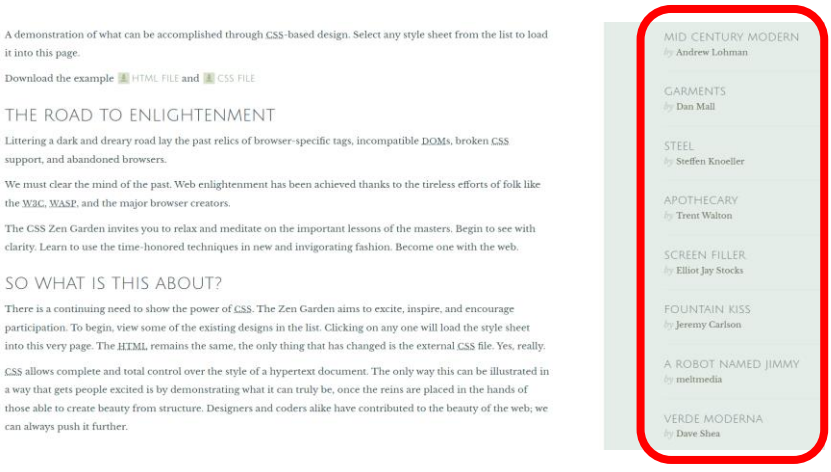
The use of CSS to control the appearance of a webpage is based on the principle of **separation of concerns**, where a program is divided into distinct sections such that each section deals with one aspect of the program and has minimal knowledge of the other parts.

In a webpage, how content is organized (structure) and how content is displayed (presentation) are largely independent of each other. Hence the two aspects should be handled separately. In particular, the structure of a webpage is defined using HTML in one file and its presentation is defined using CSS in a separate file.

Separating these two components facilitates modification of the structure of a webpage without affecting its presentation and vice versa. Hence work teams can be responsible for each aspect without having to worry about the other. In addition, there can be specialization of the HTML and CSS languages, so that each language is more suited for its purpose.

To better understand how such separation works, go to **`http://www.csszengarden.com`** using Google Chrome.

In the webpage, select any of the designs to see how changing the CSS design can dramatically affect the appearance of a webpage.

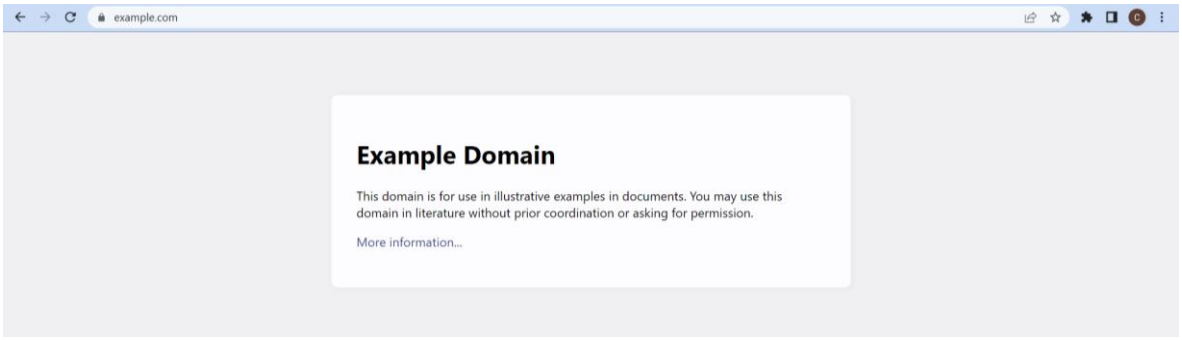


When changing to a new design, press CTRL + U to view the HTML source to verify that it remains unchanged regardless of the design.

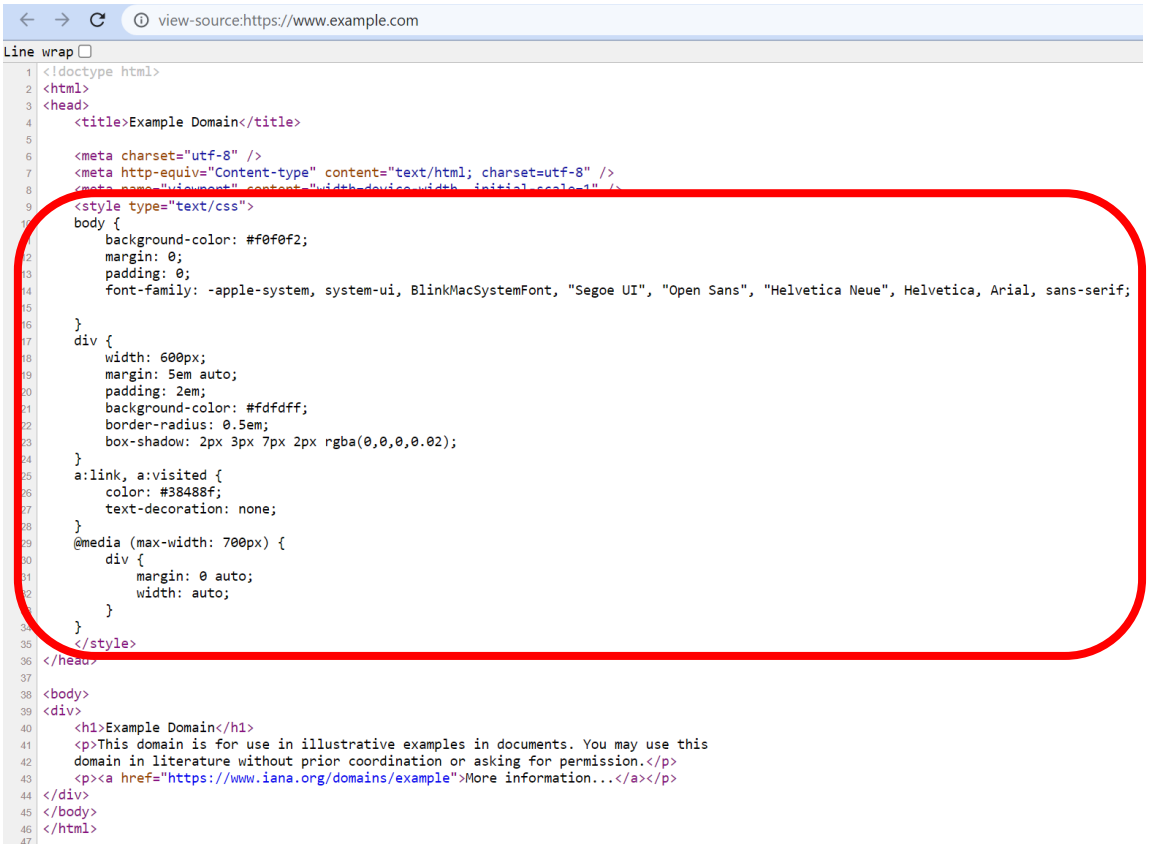
This is possible because the webpage's structure and presentation are controlled separately such that the presentation can be changed independently using CSS without modifying the HTML script that controls the structure.

2 Anatomy of a CSS Script

On Google Chrome, go to **`https://www.example.com`**.



View the HTML source of the page in Google Chrome by pressing `CTRL` + `U`.



For simplicity, this webpage has chosen to include its CSS in the first part of the HTML file within the **`<style>`**…**`</style>`** tags.

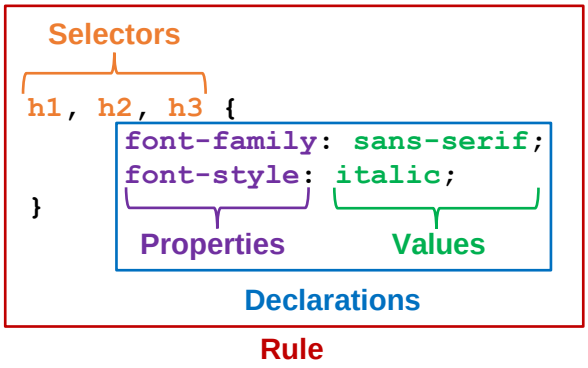
However, in general we will write our style sheet in a separate **`.css`** file and link it to a HTML file using a **`<link>`** tag. The **`<link>`** tag is a tag for a void element. Hence there is only a start tag but no end tag.

Using the CSS script for **`https://www.example.com`** as an example, we can see that the script is made up of multiple **rules**.

Each **rule** starts with one or more **selectors** separated by commas, followed by curly braces surrounding a number of **declarations**.

Each **declaration** is made of **two** parts: a **property** name and one or more **values** separated by spaces. Multiple declarations in a rule are separated by semicolons.

An example of this structure is given for a rule below.



Exercise 1

Identify all the selectors and all the properties in the following CSS script.

```
body {
      margin: 0;
      padding: 0;
}

h1, h2, #danger {
      border: 1px solid red;
}
```

Answer

Exercise 2

Which of the following scripts are valid CSS scripts? Note that **`background`** and **`color`** are valid CSS properties.

(A)
```
body (
      background: white;
)
```

(B)
```
body {
      background: white
      color: black
}
```

(C)
```
body {
      background: white;
      color: black;
}
```

(D)
```
{ color: black; }
```

(E)
```
body { color: black; }
```

Answer

3 Constructing a Sample Webpage Style Using CSS

Like HTML documents, CSS documents are basically plain text files except that they use a **`.css`** extension. In our curriculum, we shall use Notepad++ which has syntax highlighting functionality to create our CSS files.

Exercise 3

1. Open Notepad++.

2. Create a new file and save it in your working directory as **`example.css`**.

3. Enter the following CSS script.

```
h1 { color: red; font-family: sans-serif; }
p { color: blue; font-style: italic; }
```

4. Save the file again to update it with the changes.

To use a stylesheet, we need an accompanying HTML document.

Exercise 4

1. Open Notepad++.

2. Create a new file and save it as **`example.html`**.
   (The file should be saved in the same directory as **`example.css`**)

3. Enter the following HTML.

```
<!doctype html>
<html>
	<head>
		<title>CSS Example</title>
		<link rel="stylesheet" href="example.css">
	</head>
	<body>
		<h1>CSS Example</h1>
		<p>Example of using CSS to style a webpage.</p>
	</body>
</html>
```

4. Save the file again to update it with the changes.

5. Open the file **`example.html`** on the web browser.

When **Exercises 3** and **4** are done correctly, the resulting webpage will look as follows:

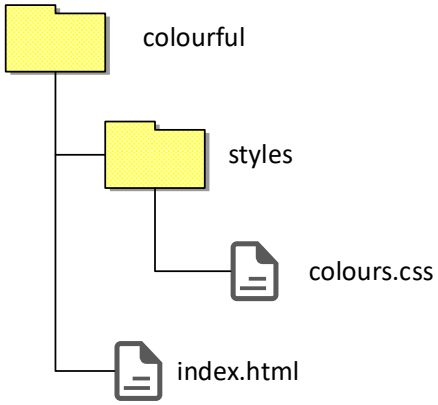
Notice that a **<link>** tag has been added under the head element of the HTML script in addition to the **<title>**…**</title>** tags. The **<link>** tag establishes a link between the HTML and CSS documents so that the CSS can be used to format the appearance of the webpage. This linkage is one type of metadata that can be contained in the head element other than the mandatory title elememt.

Within the **<link>** tag, the relationship attribute **rel** has a value of "**stylesheet**". This indicates that the hypertext reference **href** attribute of the tag will contain the relative URL of CSS file. This is normally how a style sheet is associated to a HTML document.

The use of relative URL ensures that the style sheet is always linked correctly to the HTML document even when the entire directory and its contents are moved to another location (assuming no change in directory structure after movement).

Exercise 5

(a) Create a HTML document **index.html** is styled using **colours.css**. Save the files using the following directory structure.



Use the following CSS script for **colours.css**.

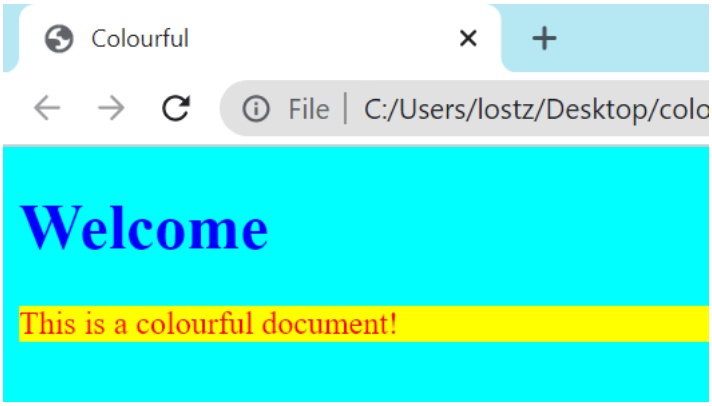
```
body { background: cyan; }
h1 { color: blue; }
p {
      background: yellow;
      color: red;
}
```

Use the following HTML script for **index.html.** You will need to establish the link between **index.html** and **colours.css** by yourself.

```
<!doctype html>
<html>
      <head>
            <title>Colourful</title>
      </head>
      <body>
            <h1>Welcome</h1>
            <p>This is a colourful document!</p>
      </body>
</html>
```

Answer

The output page will be as follows:



4 Selectors

The elements which are affected by each CSS rule are determined by the selectors at the start of that rule.

Each selector identifies a set of elements from the HTML document to be affected by the rule's declarations.

Selectors can be organized into four groups:

- element selectors
- ID selectors
- class selectors
- descendent selectors.

4.1 Element Selectors

An element selector identifies **all** the elements of a particular type from the HTML document.

To use an element selector, we simply enter the name of a tag or element type (e.g. **`body`**, **`h1`**, **`table`**, etc.).

For instance, the second rule in **`example.css`** (see **Exercise 3**) uses an element selector for paragraph **`p`** elements.

```
p { color: blue; font-style: italic; }
```

selector

Based on the above rule, all paragraph **`p`** elements on the web page will appear as blue italic text.

What happens then if we do not we do not want to select all the elements of a particular type?

In such cases, we can customize our selection by making use of two special attributes that are valid for all HTML tags: **`id`** and **`class`**.

4.2 ID Selectors

An ID selector identifies **the unique element** that has a particular value for its **`id`** attribute.

To use an ID selector, we enter a hashtag (**#**) followed immediately by the value of the required element's **`id`** attribute. Since **`id`** attributes on a web page cannot be repeated, an ID selector will always identify **exactly one** element if it exists.

Exercise 6

Create a CSS file **`id-example.css`** to style a HTML document **`id-example.html`** using the following scripts. Save both files in a folder named **`id-example`**.

The CSS file **`id-example.css`** contains CSS script that will style the element with an **`id`** of **`special`** in red font color.

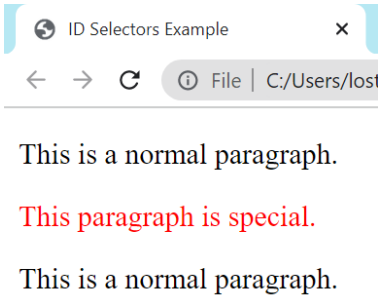
CSS file

```
#special { color: red; }
```

HTML file

```
<!doctype html>
<html>
    <head>
        <title>ID Selectors Example</title>
        <link rel="stylesheet" href="id-example.css">
    </head>
    <body>
        <p>This is a normal paragraph.</p>
        <p id="special">This paragraph is special.</p>
        <p>This is a normal paragraph.</p>
    </body>
</html>
```

If done correctly, only the second paragraph **`p`** element with an **`id`** of **`special`** is formatted in red font color.



4.3 Class Selectors

A class selector identifies **all** the elements that are associated with a particular class.

To use a class selector, we enter the dot symbol (`.`) followed immediately by the class name to be referenced.

Exercise 7

Create a CSS file **`class-example.css`** to style a HTML document **`class-example.html`** using the following scripts. Save both files in a folder named **`class-example`**.

The CSS file **`class-example.css`** contains CSS script that will style all the elements associated with the class **`info`** in silver font colour.

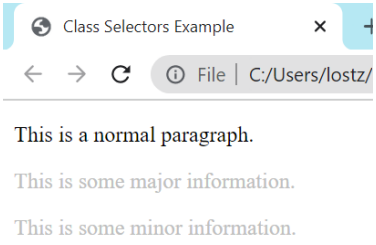
CSS file: `class-example.css`

```
.info { color: silver; }
```

HTML file: `class-example.html`

```
<!doctype html>
<html>
    <head>
        <title>Class Selectors Example</title>
        <link rel="stylesheet" href="class-example.css">
    </head>
    <body>
        <p>This is a normal paragraph.</p>
        <p class="info major">This is some major information.</p>
        <p class="info minor">This is some minor information.</p>
    </body>
</html>
```

If done correctly, the second and third paragraph **`p`** elements belonging to the **`class`** named **`info`** will be formatted in silver font color.



Note that an element can belong to more than one **`class`**.

In **Exercise 7**, the second and third paragraph **`p`** elements that belong to the **`class`** named **`info`** also belong to the **`class`** named **`major`** and the **`class`** named **`minor`** respectively. There is however no impact as they do not match any of the selectors used in the style sheet.

4.4 Use of `id` and `class` Attributes

A quick comparison of the id and class attributes is given in the table below.

`id` Attribute	**`class`** Attribute
Uniquely identifies an element in a HTML document.	Used to associate an element with one or more **`class`** i.e. an element can belong to more than one class.
Value assigned to the attribute must be unique within the document and cannot contain any whitespace.	Values assigned must be a space separated list of class names i.e there can be more than one value assigned to the class attribute and two values are separated by a space.
A webpage cannot have more than one element with the same value for the **`id`** attribute.	Multiple elements on a webpage can belong to the same **`class`**.

4.5 Descendent Selectors

Sometimes, element selectors, ID selectors and class selectors are insufficient. It may be necessary to select an element only if the element has a parent element that matches another selector. This can be achieved using the descendent selector.

To use a descendent selector, separate any two selectors using a space. This will cause the rule to apply only to elements matching the selector on the right **and** have a parent element matching the selector on the left.

Exercise 8

Create a CSS file **`descendent-example.css`** to style a HTML document **`descendent-example.html`** using the following scripts. Save both files in a folder named **`descendent-example`**.

The CSS file **`descendent-example.css`** contains CSS script that will style all the elements within the italic tags **`<i>`**…**`</i>`** nested within the paragraph **`<p>`**…**`</p>`** tags in red color font. Elements within the italic tags **`<i>`**…**`</i>`** but are not nested within the paragraph **`<p>`**…**`</p>`** tags will not be styled.

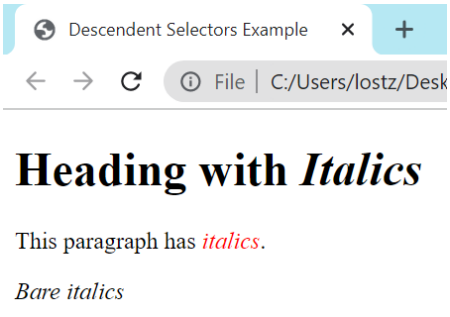
CSS file: `descendent-example.css`

```
p i { color: red; }
```

HTML file: `class-example.html`

```
<!doctype html>
<html>
    <head>
        <title>Descendent Selectors Example</title>
        <link rel="stylesheet" href="descendent-example.css">
    </head>
    <body>
        <h1>Heading with <i>Italics</i></h1>
        <p>This paragraph has <i>italics</i>.</p>
        <i>Bare italics</i>
    </body>
</html>
```

If done correctly, the word “italics” in the second paragraph will be styled in red font color.



Observe that all other italic **`i`** elements are not formatted in red font color as they do not match the specific requirements of the selector, which requires the italic **`i`** element to be a descendent of a paragraph **`p`** element i.e. the italic **`i`** element needs to be nested within the paragraph **`p`** element.

4.6 Conflicts in Declarations

When using CSS, there may be instances where an element may be selected by more than one rule in the style sheet.

If these rules have conflicting declarations, the rule with a selector that is most specific to the element has priority.

The CSS language has a pre-defined algorithm for calculating the specificity of a selector.

However, we can avoid worrying about specificity by crafting our style sheets to avoid rule conflicts in the first place.

Exercise 9

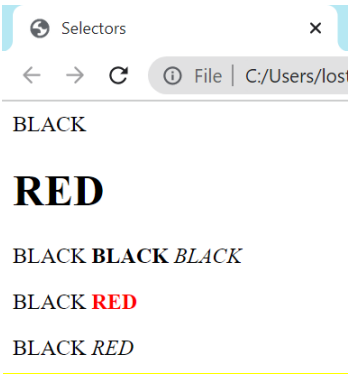
Consider the following HTML file **`selectors.html`** that contains the script below:

```
<!doctype html>
<html>
    <head>
        <title>Selectors</title>
        <link rel="stylesheet" href="selectors.css">
    </head>
    <body>
        BLACK
        <h1>RED</h1>
        <p>BLACK <b>BLACK</b> <i>BLACK</i></p>
        <p id="test1">BLACK <b>RED</b></p>
        <p class="test2">BLACK <i>RED</i></p>
    </body>
</html>
```

Without changing the HTML script, add a CSS file **`selectors.css`** in the same directory such that only the text "**`RED`**" is in red font color.

Answer

If done correctly, the output file will be as follows:



5 CSS Properties

Now that we know how CSS determines which elements are affected by each rule, we will need to next understand how various properties can be set by rule declarations.

These properties can be classified into three major groups:
- common properties,
- box model properties
- typography properties

5.1 Common Properties

A number of properties can be used with any element.

(A) Specifying Colors for Background and Text

For instance, in many of the CSS examples and exercises so far, we have been using the **`background`** and **`color`** properties to set the background and text color of elements respectively.

The following are some of the color names that can be used with the **`background`** and **`color`** properties:

red	orange	yellow	green	blue	purple
black	gray	silver	white	transparent	
				(no color)	

RGB Representation

If a desired color does not match any of the above names, we can also specify a color in terms of its red, green and blue components.

Each component is expressed as an integer between **`0`** to **`255`** (inclusive) and the color is written as **`rgb(R, G, B)`** where **`R`** is the red component integer, **`G`** is the green component integer and **`B`** is the blue component integer.

For example, the following CSS scripts sets the page background to a shade of pale yellow:

```
body { background: rgb(255, 255, 128); }
```

Hexadecimal Representation

The same color can be expressed as three hexadecimal numbers of 2 digits each (including a leading zero if required).

The color can thus be written as **`#RRGGBB`**, where **`RR`** is the red component in hexadecimal, **`GG`** is the green component in hexadecimal and **`BB`** is the blue component in hexadecimal.

For example, the shade of pale yellow shown previously in RGB representation can also be written as:

```
body { background: #ffff80; }
```

where the hexadecimals **`ff`** and 80 represent the denary values **`255`** and **`80`** respectively.

Also note that hexadecimal digits are not case sensitive.

For convenience, if each of the three hexadecimal numbers is made of repeated digits (e.g. **00**, **11**, **22**, ..., **FF**), then the color can be shortened to **#RGB**, where **R** is the repeated hexadecimal digit for red, **G** is the repeated hexadecimal digit for green and **B** is the repeated hexadecimal digit for blue.

For example, while **#FFFF80** cannot be shortened, a color such as **#00FFCC** can be shortened to **$0FC**.

(B) Creating a Background by Repeating an Image

Besides colors, we can repeat an image to form the background by using **url(PATH)**, where **PATH** is the relative URL of the background image.

For instance, the following CSS script uses pattern.png (which is located in the same folder with the style sheet) as a repeating background image for the page:

```
body { background: url(pattern.png); }
```

Exercise 10

Consider the following HTML file **repeated-image.html** that contains the script below:

```
<!doctype html>
<html>
    <head>
        <title>Repeated Image</title>
    </head>
    <body>
        <h1>HELLO WORLD!</h1>
    </body>
</html>
```

The HTML file is saved in the folder **image-repeat**.

(a) Write a CSS script that will allow the image **sample.png** to be displayed repeatedly to form the background of **repeated-image.html**. Save the CSS script as **repeated-image.css** in a sub-folder of **image-repeat** named **styles**.

You will need to select an image to be used as **sample.png** (the extension may be changed to **.jpg** or **.gif** depending on the format of the image you have selected). Save the image in the sub-folder **styles**.

(b) Establish the link between the HTML and CSS files.

Answer

If done correctly, the webpage should look similar to the following:

(C) The `display` Property

Hiding elements

All elements have a **`display`** property that can be used to hide the element by setting its value to **`none`**.

Exercise 11

Create a CSS file **`hide-display.css`** to style a HTML document **`hide-display.html`** using the following scripts. Save the HTML file in the folder named **`hide-display`** and the CSS file in its sub-folder **`styles`**.

The CSS file **`hide-display.css`** contains CSS script that will hide the element with an **`id`** of **`special`**.

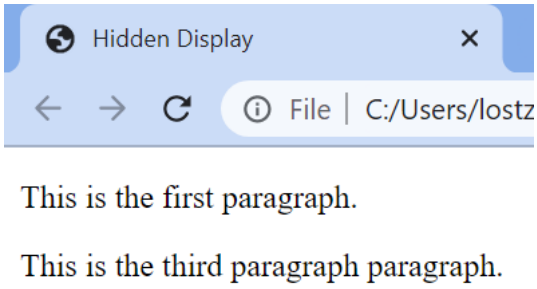
CSS file

```
#special { display: none; }
```

HTML file

```
<!doctype html>
<html>
    <head>
        <title>Hidden Display</title>
        <link rel="stylesheet" href="styles/hide-display.css">
    </head>
    <body>
        <p>This is the first paragraph.</p>
        <p id="special">The second paragraph is hidden.</p>
        <p>This is the third paragraph paragraph.</p>
    </body>
</html>
```

If done correctly, the webpage will display as follows, where the second paragraph is not displayed.



Block and Inline Appearance

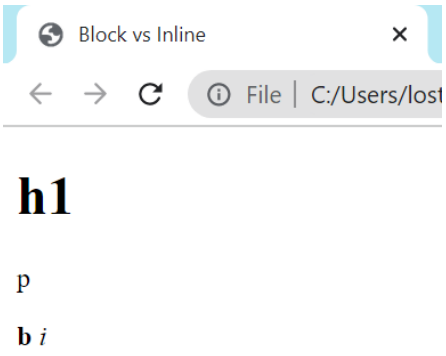
You may have noticed that some HTML tags such as **<h1>…</h1>** and ** … ** always start on a new line and force the following element to also start on a new line.

On the other hand, tags such as **…** and **…** do not.

This is because tags such as **<h1>…</h1>** and ** … ** have a block appearance by default while tags such as **…** and **…** have an inline appearance by default.

Let us understand what this means:

HTML Script	Result

<!doctype html>
<html>
  <head>
  <title>Block vs Inline</title>
  </head>
  <body>
   h1
  p
  b
  i
  </body>
</html>
	

From the illustration above, we can see that tags will a block appearance appear as separate lines while tags with an inline appearance appear on the same line.

The **display** property can be used to override default block and inline appearances.

For instance, the following CSS script changes all the elements to use a block appearance instead.

```
h1, p, b, i { display: block; }
```

Similarly, the following sample CSS script changes all the elements to use an inline appearance instead.

```
h1, p, b, i { display: inline; }
```

To better understand, let us take a look at the output when the **display** property is set to the values **block** and **inline** respectively in the following exercises.

Exercise 12

Create HTML and CSS files using the following scripts. Save the HTML script as **`block-appearance.html`** in a folder **`block-appearance`** and the CSS file as **`block-appearance.css`** in its sub-folder **`styles`**.

Link the CSS file to the HTML file so that the output will be styled in block appearance format.

CSS Script

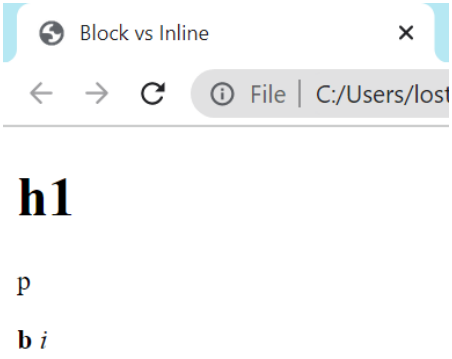
```
h1, p, b, i { display: block; }
```

HTML Script

```
<!doctype html>
<html>
    <head>
        <title>All Block</title>
    </head>
    <body>
        <h1>h1</h1>
        <p>p</p>
        <b>b</b>
        <i>i</i>
    </body>
</html>
```

Answer

The expected output is as follows:



Exercise 13

Create HTML and CSS files using the following scripts. Save the HTML script as **inline-appearance.html** in a folder **inline-appearance** and the CSS file as **inline-appearance.css** in its sub-folder **styles**.

Link the CSS file to the HTML file so that the output will be styled in inline appearance format.

CSS Script

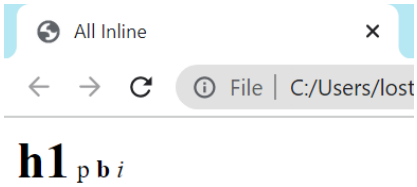
```
h1, p, b, i { display: inline; }
```

HTML Script

```
<!doctype html>
<html>
    <head>
        <title>All Inline</title>
    </head>
    <body>
        <h1>h1</h1>
        <p>p</p>
        <b>b</b>
        <i>i</i>
    </body>
</html>
```

Answer

The expected output is as follows:



Exercise 14

For each tag in the table below, determine if it creates a block or inline element by default. (You may create relevant HTML scripts to conduct a test where necessary.)

Tag	Default Appearance
`<h1>`	Block / Inline
`<h2>`	Block / Inline
`<h3>`	Block / Inline
`<p>`	Block / Inline
`<b>`	Block / Inline
`<i>`	Block / Inline
`<a>`	Block / Inline
`<img>`	Block / Inline
`<table>`	Block / Inline

(D) The and Tags

Sometimes, we may need a generic block or inline element to represent some structural features in a HTML document that the built-in HTML tags do not provide.

The ** ** and **** tags can be used in the HTML script to create block and inline elements respectively for this purpose. These tags are generally used in combination with the **id** and **class** attributes to help clarify their purpose.

For instance, the following HTML document uses the ** ** and **** tags to enable the CSS script to correctly style the document. The CSS script will set the background of the block elements that belongs to the class **rule** to sliver and the font color of the inline elements that belong to the class **keyword** to red.

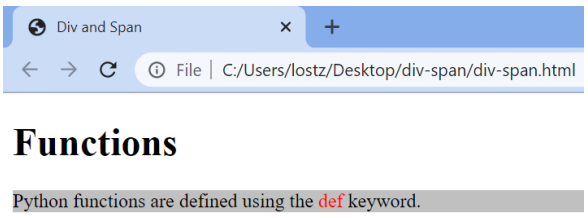
HTML Script

```
<!doctype html>
<html>
    <head>
        <title>Div and Span</title>
        <link rel="stylesheet" href="div-span.css">
    </head>
    <body>
        <h1>Functions</h1>
        <div class="rule">
            Python functions are defined using the
            <span class="keyword">
                def
            </span>
            keyword.
        </div>
    </body>
</html>
```

CSS Script

```
.rule { background: silver; }
.keyword { color: red; }
```

Expected Output

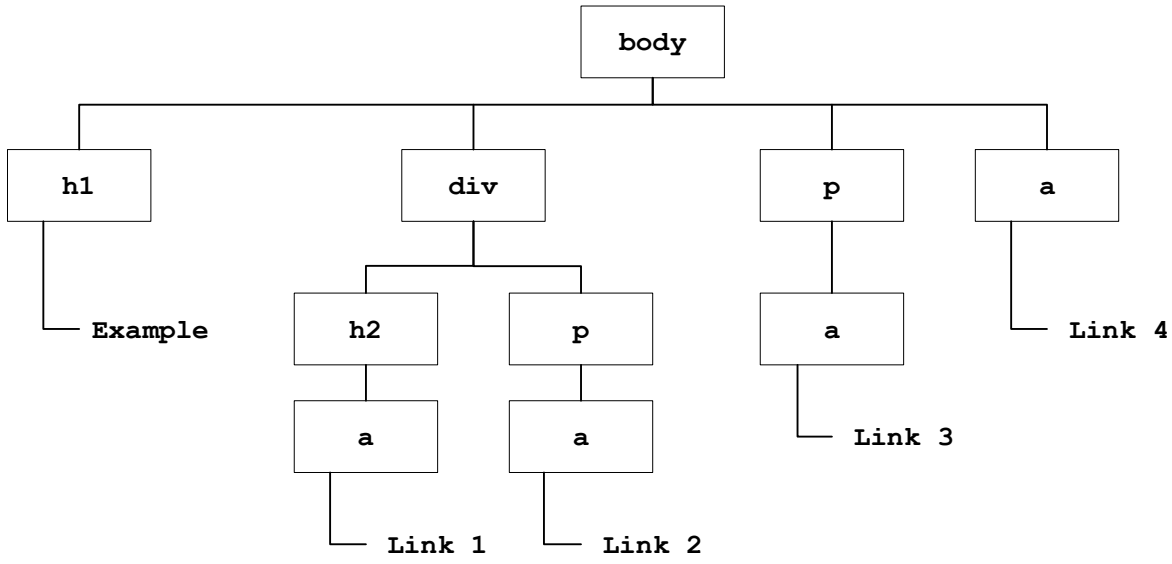


The ** ** tag can be used to create block(s) of elements to enable the selectors in CSS to work correctly to achieved the required styling.

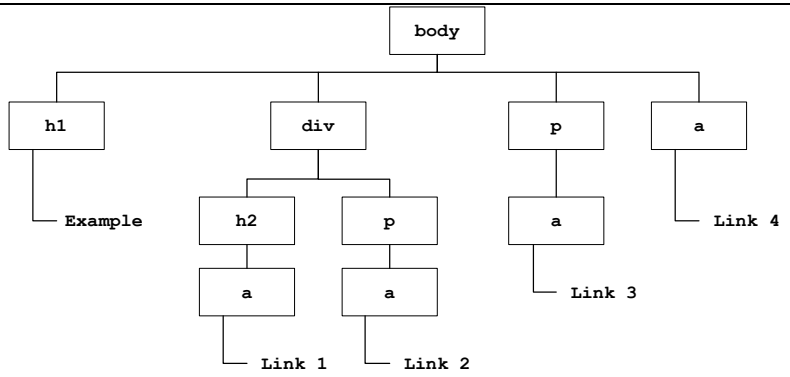
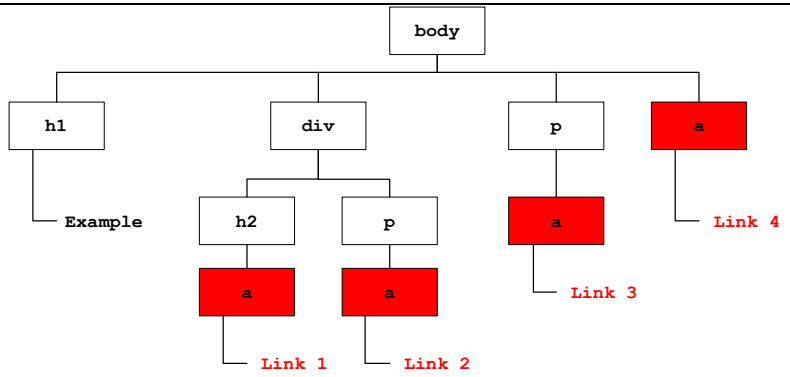
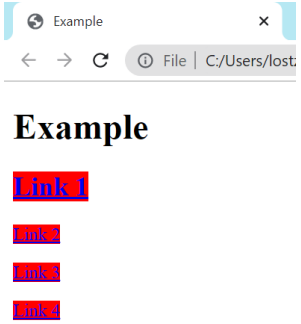
Consider the following HTML document **multiple-example.html** linked to different CSS scripts.

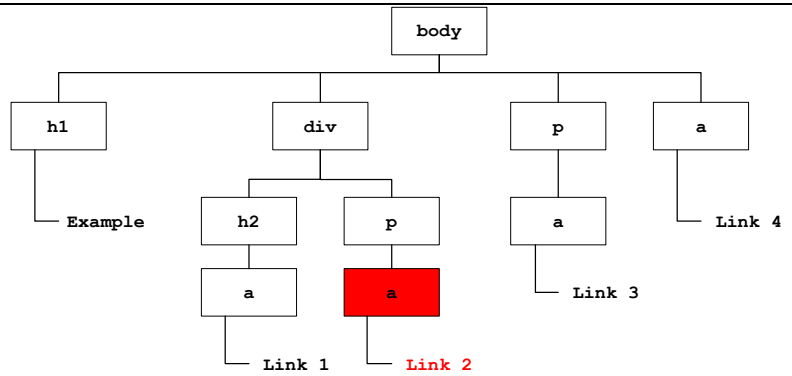
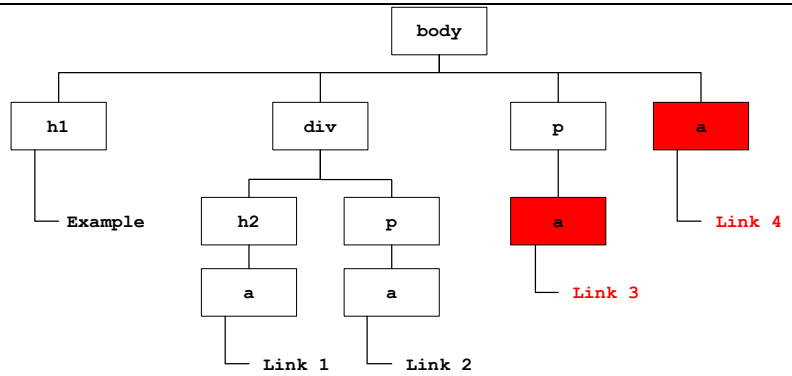
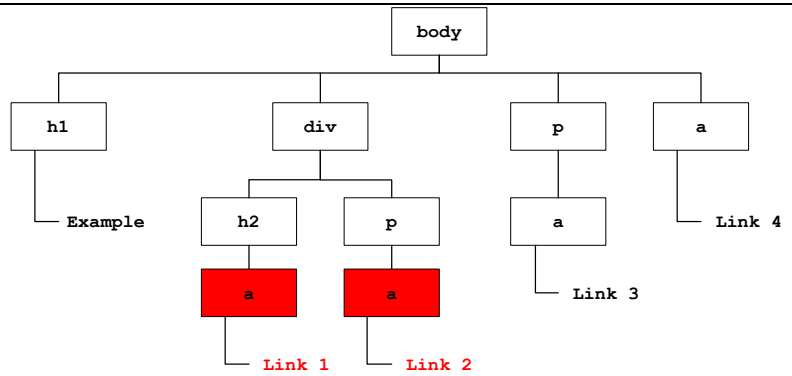
```
<!doctype html>
<html>
    <head>
        <title>Example</title>
        <link rel="stylesheet" href="multiple-example.css">
    </head>
    <body>
        <h1>Example</h1>
            <div>
                <h2>
                    <a id="link1" href="link1.html">Link 1</a>
                </h2>
                <p>
                    <a id="link2" href="link2.html">Link 2</a>
                </p>
            </div>
            <p>
                <a class="simple" href="link3.html">Link 3</a>
            </p>
            <a class="simple" href="link4.html">Link 4</a>
    </body>
</html>
```

The above HTML can be illustrated using the following tree structure of elements.



The following table provides an analysis of how the elements are affected based on the different selectors used in the CCS script of the CSS file **`multiple-example.css`** that is linked to the above HTML document.

CSS Script	Affected Elements	Sample Output
No style sheet		
`a { background: red; }`		

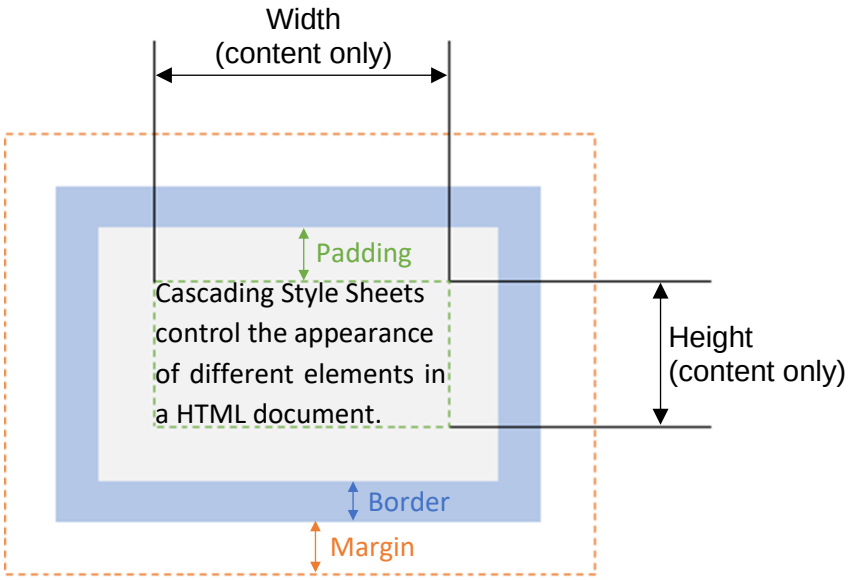
CSS Script	Affected Elements	Sample Output
`#link2 { background: red; }`		
`.simple { background: red; }`		
`div a { background: red; }`		

5.2 Box Model Properties

Besides background and text colour, block elements have a number of additional properties that can be specified.

In particular, the **`margin`**, **`border`**, **`padding`**, **`width`** and **`height`** properties determine how the “box” surrounding the element is sized and displayed.

This **box model** is illustrated below:



The various box model properties are summarized in the following table.

Property	Description
`border`	Specifies thickness of the optionally colored border around the element.
`margin`	Specifies thickness of the transparent space surrounding the border.
`padding`	Specifies thickness of the space between the content and the border that is filled with the element's background color or pattern.
`width`	Specifies the width of the contents of the element, regardless of the surrounding margin, border and padding.
`height`	Specifies the height of the contents of the element, regardless of the surrounding margin, border and padding

When setting a box's width and height or the thickness of its margin, border and padding, we must specify a unit of measurement.

One common unit of measurement is pixels, represented by the **px** suffix.

For instance, to set the padding of paragraph **p** elements to 10 pixels, we would use the following CSS script:

```
p {
  background: silver;
  padding: 10px;
}
```

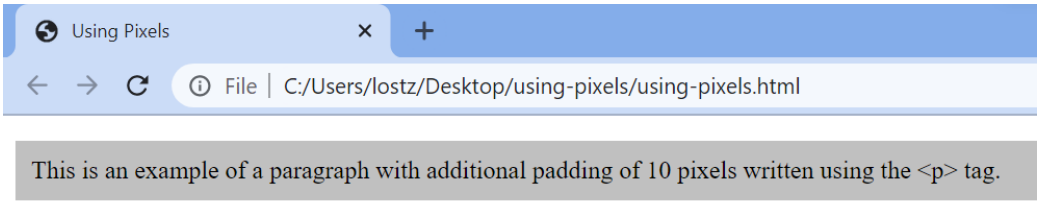
Exercise 15

Create a HTML file **using-pixels.html** and a CSS file **using-pixels.css** that makes use of the above CSS script. Save the HTML file in a folder named **using-pixels** and the CSS script in its sub-folder **styles**.

HTML Script

```
<!doctype html>
<html>
    <head>
        <title>Using Pixels</title>
        <link rel="stylesheet" href="styles/using-pixels.css">
    </head>
    <body>
        <p>
            This is an example of a paragraph with additional
            padding of 10 pixels written using the &lt;p&gt; tag.
        </p>
    </body>
</html>
```

If done correctly, the output webpage is as follows:



To specify that a border should be drawn with a solid color, we use

- the value of a thickness, followed by a space,
- followed by the word "**solid**", then another space,
- and finally the color we wish to use for the border.

For instance, the following CSS gives paragraph **p** elements a solid red border of 5 pixels thickness.

```
p { border: 5px solid red; }
```

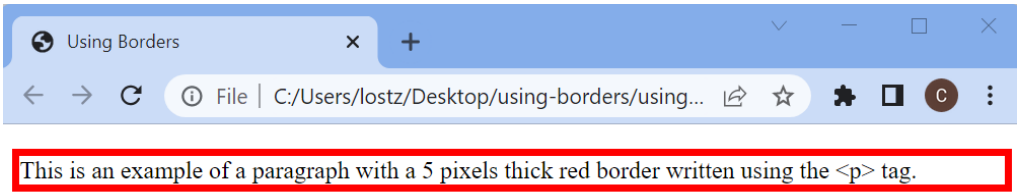
Exercise 16

Create a HTML file **using-borders.html** and a CSS file **using-borders.css** that makes use of the above CSS script. Save the HTML file in a folder named **using-borders** and the CSS script in its sub-folder **styles**.

HTML Script

```
<!doctype html>
<html>
    <head>
        <title>Using Borders</title>
        <link rel="stylesheet" href="styles/using-borders.css">
    </head>
    <body>
        <p>
            This is an example of a paragraph with a 5 pixels
            thick red border written using the &lt;p&gt; tag.
        </p>
    </body>
</html>
```

If done correctly, the output webpage is as follows:



By default, the **margin**, **border** and **padding** properties control the appearance for all four sides of the element's box. However, we can append **-bottom**, **-left**, **-top** or **-right** to any of these properties so that we control the appearance for only one side of the box.

For instance, the following CSS script shows how the **border-bottom**, **border-left**, **border-top** and **border-right** properties can be used to draw a line on only one side of a paragraph **p** element.

```
p {
  border-bottom: 2px solid red;
  border-left: 2px solid blue;
  border-top: 2px solid green;
  border-right: 2px solid gray;
}
```

Note that note all 4 borders need to be drawn concurrently. You may choose to a borderline on only one side e.g.

```
p { border-bottom: 2px solid red; }
```

Exercise 17

Create a HTML file with the name **different-borders.html** and a CSS file with the name **different-borders.css**. Save the HTML file in a folder named **different-borders** and the CSS script in its sub-folder **styles**.

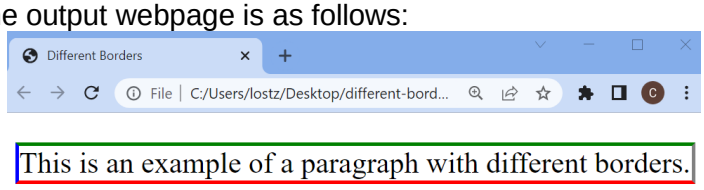
HTML Script

```
<!doctype html>
<html>
   <head>
      <title>Different Borders</title>
      <link rel="stylesheet" href="styles/different-borders.css">
   </head>
   <body>
      <p>
         This is an example of a paragraph with different borders.
      </p>
   </body>
</html>
```

CSS Script

```
p {
  border-bottom: 2px solid red;
  border-left: 2px solid blue;
  border-top: 2px solid green;
  border-right: 2px solid gray;
}
```

If done correctly, the output webpage is as follows:



Finally, if a box's width is less than the document's width, we can center the box horizontally by setting its **margin-left** and **margin-right** properties to **auto**.

Exercise 18

Create a HTML file with the name **centered-box.html** and a CSS file with the name **centered-box.css**. Save the HTML file in a folder named **centered-box** and the CSS script in its sub-folder **styles**.

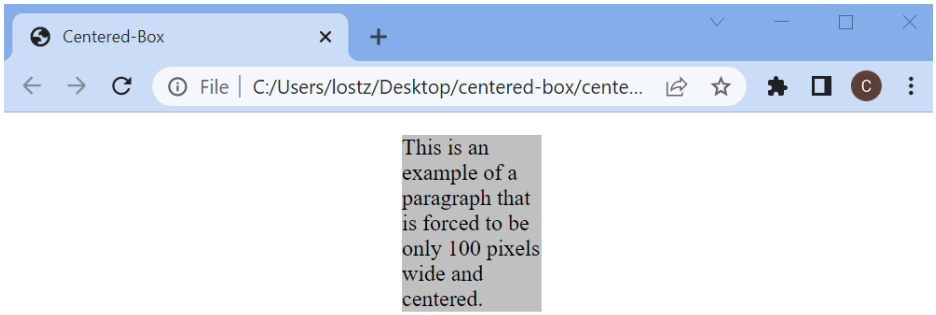
HTML Script

```
<!doctype html>
<html>
   <head>
      <title>Centered-Box</title>
      <link rel="stylesheet" href="styles/centered-box.css">
   </head>
   <body>
      <p>
         This is an example of a paragraph that is forced to be
         only 100 pixels wide and centered.
      </p>
   </body>
</html>
```

CSS Script

```
p {
    background: silver;
    margin-left: auto;
    margin-right: auto;
    width: 100px;
}
```

If done correctly, the output webpage is as follows:



Exercise 19

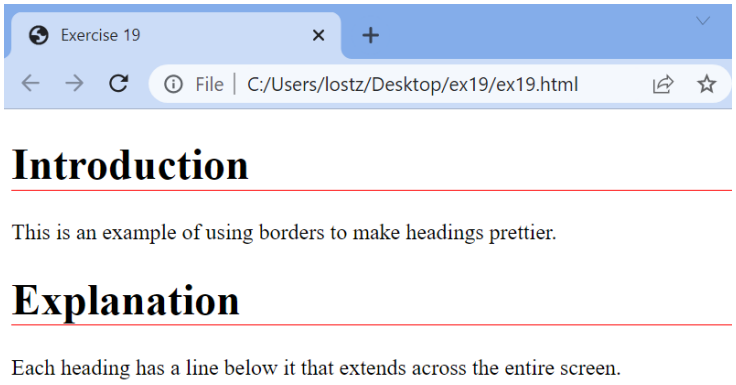
For the HTML script below, create a CSS file so that the completed web page appears as closely to the required appearance as possible.

HTML Script : `ex19.html` saved in folder `ex19`

```
<!doctype html>
<html>
    <head>
        <title>Exercise 19</title>
        <link rel="stylesheet" href="styles/ex19.css">
    </head>
    <body>
        <h1>Introduction</h1>
        <p>
            This is an example of using borders to make headings
            prettier.
        </p>
        <h1>Explanation</h1>
        <p>
            Each heading has a line below it that extends across
            the entire screen.
        </p>
    </body>
</html>
```

Required Output

If done correctly, the output webpage is as follows:



Answer

Exercise 20

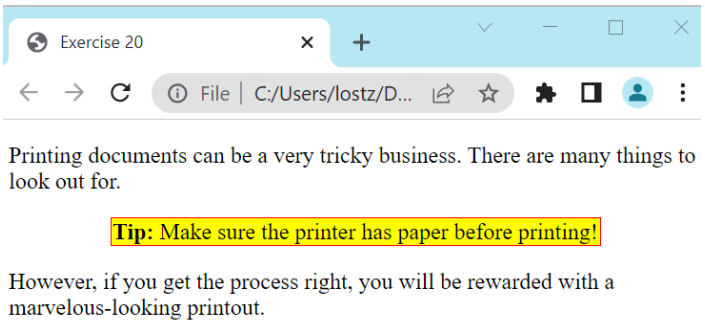
For the HTML script below, create a CSS file so that the completed web page appears as closely to the required appearance as possible.

HTML Script : `ex20.html` saved in folder `ex20`

```
<!doctype html>
<html>
    <head>
        <title>Exercise 20</title>
        <link rel="stylesheet" href="styles/ex20.css">
    </head>
    <body>
        <p>
            Printing documents can be a very tricky business.
            There are many things to look out for.
        </p>
        <p class="tip">
            <b>Tip:</b> Make sure the printer has paper before
            Printing!
        </p>
        <p>
            However, if you get the process right, you will be
            rewarded with a marvelous-looking printout.
        </p>
    </body>
</html>
```

Required Output

If done correctly, the output webpage is as follows:



Answer

Exercise 21

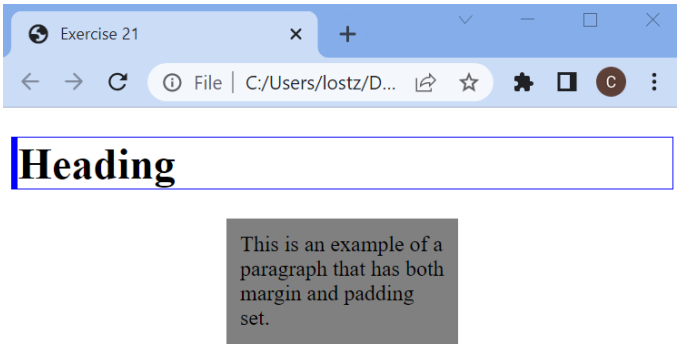
For the HTML script below, create a CSS file so that the completed web page appears as closely to the required appearance as possible.

HTML Script : `ex21.html` saved in folder `ex21`

```
<!doctype html>
<html>
    <head>
        <title>Exercise 21</title>
        <link rel="stylesheet" href="styles/ex21.css">
    </head>
    <body>
        <h1>Heading</h1>
        <p>
            This is an example of a paragraph that has both margin
            and padding set.
        </p>
    </body>
</html>
```

Required Output

If done correctly, the output webpage is as follows:



Answer

5.3 Typography Properties

Besides controlling the size and spacing of box elements, how text in each element is displayed can also be specified using the **`font-family`**, **`font-size`**, **`font-style`**, **`font-weight`**, **`text-align`** and **`text-decoration`** properties.

Font Family

The **`font-family`** property specifies which typeface is used to display the text.

Its value is most often a comma-separated list of font names, ending with either **`serif`** or **`sans-serif`**.

Note that if a font name has spaces, it must be enclosed in quotes (e.g., "Times New Roman").

The browser will use the first font in the list that is installed. However, if none of the named fonts are installed, it will fall back to using a generic **`serif`** or **`sans-serif`** font instead.

A **`serif`** font such as "Times New Roman" has lines extending from the ends of each letter stroke. Such fonts are traditionally used for long pieces of printed text.

A **`sans-serif`** font such as "Arial", however, does not have these additional lines.

Font Size

The **`font-size`** property can be used to specify text size in pixels **`px`**.

Font Style

The **`font-style`** property specifies whether an italic font is used. The most common values for this property are **`normal`** and **`italic`**.

Font Weight

The **`font-weight`** property specifies whether a bold font is used. The most common values for this property are **`normal`** and **`bold`**.

Exercise 22

Create a HTML script to display different font properties. Save the file as **fontppty.html** in the folder **fontppty**.

Use the CSS script provide to style the HTML document. Save the script as **fontppty.css** in the subfolder **styles**.

HTML Script

```
<!doctype html>
<html>
    <head>
        <title>Font Properties</title>
        <link rel="stylesheet" href="styles/fontppty.css">
    </head>
    <body>
        <p class="para1">
            This is a paragraph in serif font family.
        </p>
        <p class=" para2">
            This is a paragraph in sans-serif font family.
        </p>
        <p class=" para3">
            This is a paragraph in bold.
        </p>
        <p class=" para4">
            This is a paragraph in italics.
        </p>
        <p class=" para5">
            This is a paragraph with font size 24px.
        </p>
        <p class=" para6">
            This is a paragraph with font size 48px.
        </p>
    </body>
</html>
```

CSS Script

```
.para1 { font-family: serif; }
.para2 { font-family: sans-serif; }
.para3 { font-weight: bold; }
.para4 { font-style: italic; }
.para5 { font-size: 24px; }
.para6 { font-size: 48px; }
```

If done correctly, the output will be as follows.

Text Alignment

The **text-align** property specifies how the text is aligned in its box. The most common values for this property are **left**, **center**, **right** and **justify**.

Text Decoration

Finally, the **text-decoration** property specifies whether additional elements of the font are displayed. The most common values of this property are **none**, **underline** and **line-through**.

The **line-through** value causes text to appear cancelled or struck through using a horizontal line.

Exercise 23

Create a HTML script to display different font properties. Save the file as **textppty.html** in the folder **textppty**.

Use the CSS script provide to style the HTML document. Save the script as **textppty.css** in the subfolder **styles**.

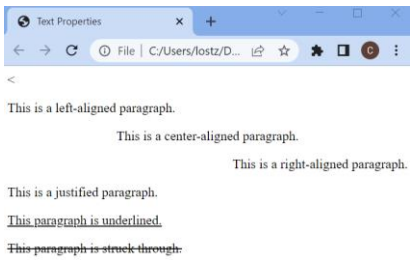
HTML Script

```
<!doctype html>
<html>
    <head>
        <title>Text Properties</title>
        <link rel="stylesheet" href="styles/textppty.css">
    </head>
    <body>
        <p class="para1">This is a left-aligned paragraph.</p>
        <p class=" para2">This is a center-aligned paragraph.</p>
        <p class=" para3">This is a right-aligned paragraph.</p>
        <p class=" para4">This is a justified paragraph.</p>
        <p class=" para5">This paragraph is underlined.</p>
        <p class=" para6">This paragraph is struck through.</p>
    </body>
</html>
```

CSS Script

```
.para1 { text-align: left; }
.para2 { text-align: center; }
.para3 { text-align: right; }
.para4 { text-align: justify; }
.para5 { text-decoration: underline; }
.para6 { text-decoration: line-through; }
```

If done correctly, the output will be as follows.



Exercise 24

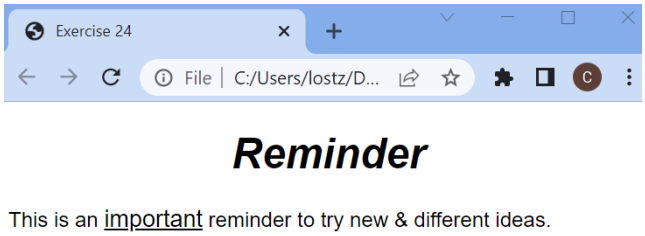
For the HTML script below, create a CSS file so that the completed web page appears as closely to the required appearance as possible.

HTML Script : `ex24.html` saved in folder `ex24`

```
<!doctype html>
<html>
    <head>
        <title>Exercise 24</title>
        <link rel="stylesheet" href="styles/ex24.css">
    </head>
    <body>
        <h1>Reminder</h1>
        <p>
            This is an <span class="expanded">important</span>
            reminder to try new &amp; different ideas.
        </p>
    </body>
</html>
```

Required Output

If done correctly, the output webpage is as follows:



Answer

Exercise 25

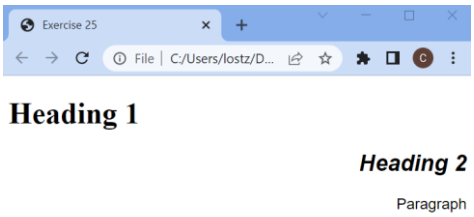
For the HTML script below, create a CSS file so that the completed web page appears as closely to the required appearance as possible.

HTML Script : `ex25.html` saved in folder `ex25`

```
<!doctype html>
<html>
    <head>
        <title>Exercise 25</title>
        <link rel="stylesheet" href="styles/ex25.css">
    </head>
    <body>
        <h1>Heading 1</h1>
        <h2>Heading 2</h2>
        <p>Paragraph</p>
    </body>
</html>
```

Required Output

If done correctly, the output webpage is as follows:



Answer

Exercise 26

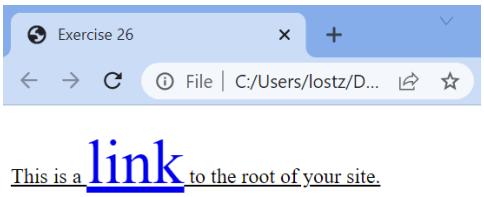
For the HTML script below, create a CSS file so that the completed web page appears as closely to the required appearance as possible.

HTML Script : `ex26.html` saved in folder `ex26`

```
<!doctype html>
<html>
    <head>
        <title>Exercise 26</title>
        <link rel="stylesheet" href="styles/ex26.css">
    </head>
    <body>
        <p>This is a <a href="/">link</a> to the root of your
           site.
        </p>
    </body>
</html>
```

Required Output

If done correctly, the output webpage is as follows:



Answer