EXPERIMENT NO. 04

DATE OF PERFORMANCE:	**GRADE:**
DATE OF ASSESSMENT:	**SIGNATURE OF LECTURER/ TTA:**

AIM: Implementation of Cascading Style Sheet.

THEORY:

CSS stands for Cascading Style Sheets.

CSS is a style sheet language that describes the presentation of an HTML (or XML) document. CSS describes how elements must be rendered on screen, on paper, or in other media. 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 style sheets are stored in CSS files.

WHY USE CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS SOLVED A BIG PROBLEM:

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!

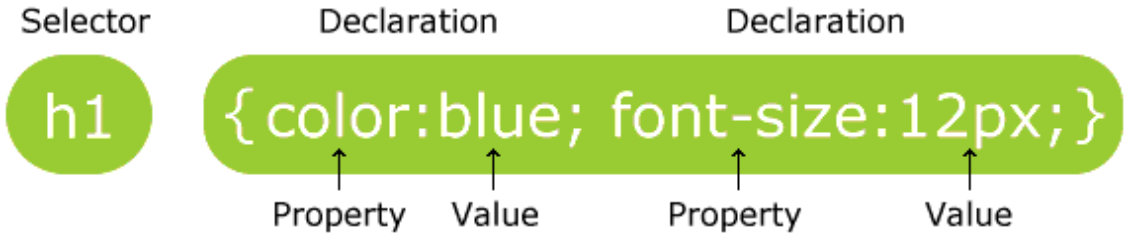
CSS SAVES A LOT OF WORK!

The style definitions are normally saved in external .css files.

With an external stylesheet file, you can change the look of an entire website by changing just one file!

CSS SYNTAX:

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.
A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all <p> elements will be center-aligned, with a red text color:

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

CSS SELECTORS:

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

The element Selector:
The element selector selects elements based on the element name.
You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

Example:

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

THE ID SELECTOR:

The id selector uses the id attribute of an HTML element to select a specific element.
The id of an element should be 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.
The style rule below will be applied to the HTML element with id="para1":

Example:

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

THE CLASS SELECTOR:

The class selector selects elements with a specific class attribute.
To select elements with a specific class, write a period (.) character, followed by the name of the class. In the example below, all HTML elements with class="center" will be red and center-aligned:

Example:

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

GROUPING SELECTORS:

If you have elements with the same style definitions, like this:

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

h2 {
   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. In the example below we have grouped the selectors from the code above:

Example:

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

CSS COMMENTS:

Comments are used to explain the code, and may help when you edit the source code at a later date. Comments are ignored by browsers.

A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

Example:

```
p {
   color: red;
   /* This is a single-line comment */
   text-align: center;
}
```

THREE WAYS TO INSERT CSS:

- **External style sheet**
- **Internal style sheet**
- **Inline style**

EXTERNAL STYLE SHEET:

With an external style sheet, you can change the look of an entire website by changing just one file! Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

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

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.
Here is how the "myStyle.css" looks:

```
body {
   background-color: lightblue;
}

h1 {
   color: navy;
   margin-left: 20px;
}
```

INTERNAL STYLE SHEET:

An internal style sheet may be used if one single page has a unique style.
Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

Example:

```
<head>
<style>
body {
   background-color: linen;
}

h1 {
   color: maroon;
   margin-left: 40px;
}
</style>
</head>
```

INLINE STYLE:

An inline style may be used to apply a unique style for a single element.
To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

Example:
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>

CASCADING ORDER:

What style will be used when there is more than one style specified for an HTML element? Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

- **Inline style (inside an HTML element)**
- **External and internal style sheets (in the head section)**
- **Browser default**

So, an inline style (inside a specific HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or a browser default value.

CSS COLORS:

Colors in CSS are most often specified by:

- **a valid color name - like "red"**
- **an RGB value - like "rgb(255, 0, 0)"**
- **a HEX value - like "#ff0000**

CSS BACKGROUNDS:

BACKGROUND COLOR:

The background-color property specifies the background color of an element.

The background color of a page is set like this:

Example

```
body {
  background-color: lightblue;
}
```

BACKGROUND IMAGE:

The background-image property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

The background image for a page can be set like this:

Example

```
body {
  background-image: url("paper.gif");
}
```

BACKGROUND IMAGE - REPEAT HORIZONTALLY OR VERTICALLY:

By default, the background-image property repeats an image both horizontally and vertically. Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Example

```
body {
  background-image: url("gradient_bg.png");
}
```

BACKGROUND IMAGE - SET POSITION AND NO-REPEAT:

Showing the background image only once is also specified by the background-repeat property:

Example

```
body {
   background-image: url("img_tree.png");
   background-repeat: no-repeat;
}
```

BACKGROUND IMAGE - FIXED POSITION:

To specify that the background image should be fixed (will not scroll with the rest of the page), use thebackground-attachment property:

Example

```
body {


   background-image: url("img_tree.png");
   background-repeat: no-repeat;
```

```
    background-position: right top;
    background-attachment: fixed;
}
```

CSS BORDERS:

CSS BORDER PROPERTIES:

The CSS border properties allow you to specify the style, width, and color of an element's border.

BORDER STYLE:

The border-style property specifies what kind of border to display.

The following values are allowed:

- **dotted - Defines a dotted border**
- **dashed - Defines a dashed border**
- **solid - Defines a solid border**
- **double - Defines a double border**
- **groove - Defines a 3D grooved border. The effect depends on the border-color value**
- **ridge - Defines a 3D ridged border. The effect depends on the border-color value**
- **inset - Defines a 3D inset border. The effect depends on the border-color value**
- **outset - Defines a 3D outset border. The effect depends on the border-color value**

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
```

BORDER WIDTH:

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
p.one {
   border-style: solid;
   border-width: 5px;
}
```

BORDER COLOR:

The border-color property is used to set the color of the four borders.

The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).

If border-color is not set, it inherits the color of the element.

Example

```
p.one {
   border-style: solid;
   border-color: red;
}
```

BORDER - INDIVIDUAL SIDES:

From the examples above you have seen that it is possible to specify a different border for each side. In CSS, there is also properties for specifying each of the borders (top, right, bottom, and left)

```
p {
   border-top-style: dotted;
   border-right-style: solid;
   border-bottom-style: dotted;
   border-left-style: solid;
}
```

CSS MARGINS:

CSS MARGINS:

The CSS margin properties are used to generate space around elements.

The margin properties set the size of the white space outside the border.

With CSS, you have full control over the margins. There are CSS properties for setting the margin for each side of an element (top, right, bottom, and left).

MARGIN - INDIVIDUAL SIDES:

CSS has properties for specifying the margin for each side of an element:

- **margin-top**
- **margin-right**
- **margin-bottom**
- **margin-left**

All the margin properties can have the following values:

- **auto - the browser calculates the margin**
- ***length* - specifies a margin in px, pt, cm, etc.**
- ***%* - specifies a margin in % of the width of the containing element**
- **inherit - specifies that the margin should be inherited from the parent element**

The following example sets different margins for all four sides of a <p> element:

Example

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

THE AUTO VALUE:

You can set the margin property to auto to horizontally center the element within its container. The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:

Example

```
div {
  width: 300px;
  margin: auto;
  border: 1px solid red;
}
```

THE INHERIT VALUE:

This example lets the left margin be inherited from the parent element:

Example

```
div.container {
  border: 1px solid red;
```

```
   margin-left: 100px;
}

p.one {
   margin-left: inherit;
}
```

MARGIN COLLAPSE:

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on horizontal margins (left and right)! Only vertical margins (top and bottom)!

Look at the following example:

Example

```
h1 {
   margin: 0 0 50px 0;
}

h2 {
   margin: 20px 0 0 0;
}
```

CSS PADDING

CSS PADDING:

The CSS padding properties are used to generate space around content.

The padding clears an area around the content (inside the border) of an element.

With CSS, you have full control over the padding. There are CSS properties for setting the padding for each side of an element (top, right, bottom, and left).

PADDING - INDIVIDUAL SIDES:

CSS has properties for specifying the padding for each side of an element:

- **padding-top**
- **padding-right**
- **padding-bottom**

- **padding-left**

All the padding properties can have the following values:

- ***length* - specifies a padding in px, pt, cm, etc.**
- ***%* - specifies a padding in % of the width of the containing element**
- **inherit - specifies that the padding should be inherited from the parent element**

The following example sets different padding for all four sides of a <p> element:

Example

```
p {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

CSS HEIGHT AND WIDTH:

SETTING HEIGHT AND WIDTH:

The height and width properties are used to set the height and width of an element.

The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block.

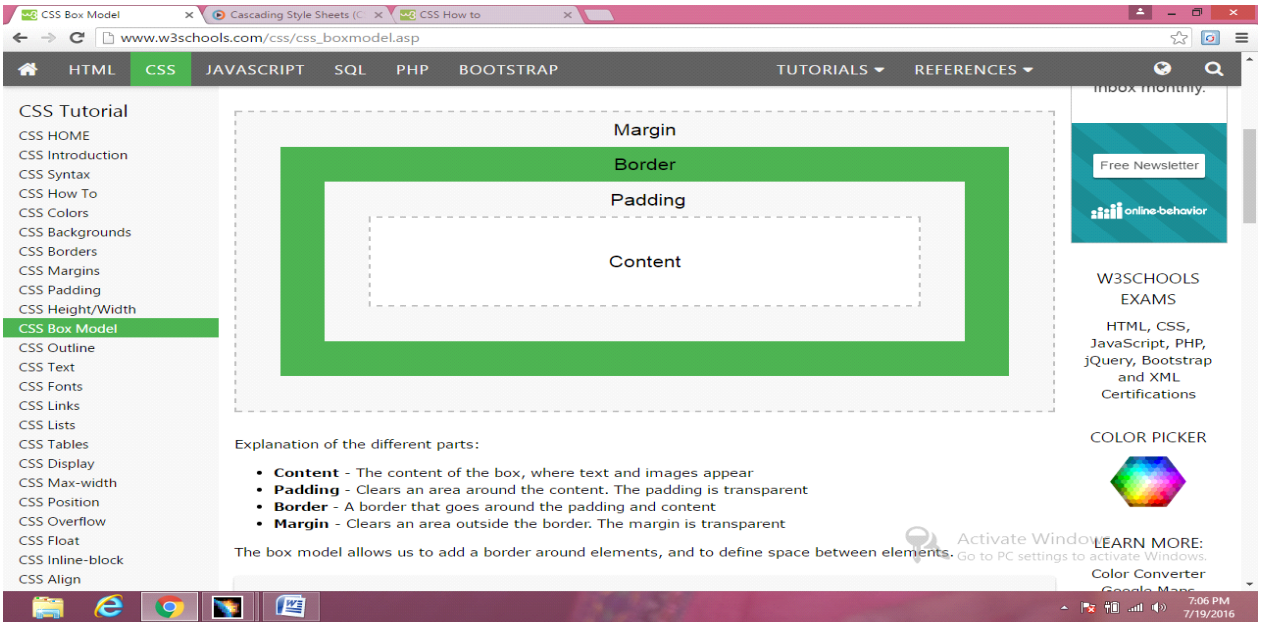
Example

```
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
```

THE CSS BOX MODEL:

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



Explanation of the different parts:

- **Content - The content of the box, where text and images appear**
- **Padding - Clears an area around the content. The padding is transparent**
- **Border - A border that goes around the padding and content**
- **Margin - Clears an area outside the border. The margin is transparent**

The box model allows us to add a border around elements, and to define space between elements.

Example

```
div {
    width: 300px;
    border: 25px solid green;
    padding: 25px;
    margin: 25px;
}
```

CSS TEXT:

TEXT COLOR:

The color property is used to set the color of the text.

Example

```
body {
    color: blue;
```

}

TEXT ALIGNMENT:

The text-align property is used to set the horizontal alignment of a text. A text can be left or right aligned, centered, or justified.

Example

```
h1 {
   text-align: center;
}
h2 {
   text-align: left;
}
```

TEXT DECORATION:

The text-decoration property is used to set or remove decorations from text.

The value text-decoration: none; is often used to remove underlines from links:

Example

```
a {
   text-decoration: none;
}
```

TEXT TRANSFORMATION:

The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

Example:

```
p.uppercase {
   text-transform: uppercase;
}
```

TEXT INDENTATION:

The text-indent property is used to specify the indentation of the first line of a text:

Example

```
p {
   text-indent: 50px;
}
```

LETTER SPACING:

The letter-spacing property is used to specify the space between the characters in a text.

Example

```
h1 {
   letter-spacing: 3px;
}
```

LINE HEIGHT:

The line-height property is used to specify the space between lines:

Example

```
p.small {
   line-height: 0.8;
}
```

TEXT DIRECTION:

The direction property is used to change the text direction of an element:

Example

```
div {
   direction: rtl;
}
```

WORD SPACING:

The word-spacing property is used to specify the space between the words in a text.

Example

```
h1 {
   word-spacing: 10px;
}
```

CSS FONTS:

The CSS font properties define the font family, boldness, size, and the style of a text.

FONT FAMILY:

The font family of a text is set with the font-family property.

The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

Example

```
p {
   font-family: "Times New Roman", Times, serif;
}
```

FONT STYLE:

The font-style property is mostly used to specify italic text.

This property has three values:

normal - The text is shown normally

italic - The text is shown in italics

oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal {
   font-style: normal;
}
```

FONT SIZE:

The font-size property sets the size of the text.

Example

```
h1 {
   font-size: 40px;
```

}

FONT VARIANT:

The font-variant property specifies whether or not a text should be displayed in a small-caps font.

Example

```
p.small {
   font-variant: small-caps;
}
```

CSS LINKS:

STYLING LINKS:

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

Example

```
a {
   color: hotpink;
}
```

The four links states are:

a:link - a normal, unvisited link

a:visited - a link the user has visited

a:hover - a link when the user mouses over it

a:active - a link the moment it is clicked

Example

```
/* unvisited link */
a:link {
   color: red;
}
/* visited link */
a:visited {
   color: green;
}
```

ADVANCED - LINK BUTTONS:

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

Example

```
a:link, a:visited {
   background-color: #f44336;
   color: white;
   padding: 14px 25px;
   text-align: center;
   text-decoration: none;
   display: inline-block;
}

a:hover, a:active {
   background-color: red;
}
```

CSS LISTS:

DIFFERENT LIST ITEM MARKERS:

The list-style-type property specifies the type of list item marker. The following example shows some of the available list item markers:

Example

```
ul.a {
   list-style-type: circle;
}
```

AN IMAGE AS THE LIST ITEM MARKER:

The list-style-image property specifies an image as the list item marker:

Example

```
ul {
   list-style-image: url('sqpurple.gif');
```

}

POSITION THE LIST ITEM MARKERS:

The list-style-position property specifies whether the list-item markers should appear inside or outside the content flow:

Example

```
ul {
   list-style-position: inside;
}
```

CSS TABLES:

Property Description

border: Sets all the border properties in one declaration

border-collapse: Specifies whether or not table borders should be collapsed

border-spacing: Specifies the distance between the borders of adjacent cells

caption-side: Specifies the placement of a table caption

empty-cells: Specifies whether or not to display borders and background on empty cells in a table

table-layout: Sets the layout algorithm to be used for a table

THE DISPLAY PROPERTY:

The display property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Example

```
li {
   display: inline;
}
span {
   display: block;
}
```

USING WIDTH, MAX-WIDTH AND MARGIN: AUTO;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the width of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to auto, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins.

Example

```
div.ex1 {
  width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}

div.ex2 {
  max-width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}
```

CSS LAYOUT - THE POSITION PROPERTY:

The position property specifies the type of positioning method used for an element (static, relative, fixed or absolute).

THE POSITION PROPERTY:

The position property specifies the type of positioning method used for an element.

There are four different position values:

- **static**
- **relative**
- **fixed**
- **absolute**

Example:

```
div.static {
  position: static;
  border: 3px solid #73AD21;
}
```

```
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

OVERLAPPING ELEMENTS:

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

Example

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

CSS OVERFLOW:

The CSS overflow property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

The overflow property has the following values:

- **visible - Default. The overflow is not clipped. It renders outside the element's box**
- **hidden - The overflow is clipped, and the rest of the content will be invisible**
- **scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content**
- **auto - If overflow is clipped, a scrollbar should be added to see the rest of the content**

CSS LAYOUT - FLOAT AND CLEAR:

The float property specifies whether or not an element should float.

The clear property is used to control the behavior of floating elements.

Example:

```
img {
  float: right;
  margin: 0 0 10px 10px;
}
```

CSS PSEUDO-CLASSES:

WHAT ARE 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**

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;
}
```

HOVER ON <DIV>:

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):

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

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

WHAT ARE PSEUDO-ELEMENTS?

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- **Style the first letter, or line, of an element**
- **Insert content before, or after, the content of an element**

Syntax

The syntax of pseudo-elements:

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

The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all <p> elements:

Example:

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

CSS OPACITY / TRANSPARENCY:

TRANSPARENT IMAGE:

The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent:

Example

```
img {
   opacity: 0.5;
   filter: alpha(opacity=50); /* For IE8 and earlier */
}

img:hover {
   opacity: 1.0;
   filter: alpha(opacity=100); /* For IE8 and earlier */
}
```

CSS NAVIGATION BAR:

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:

Example

```
<ul>
  <li><a href="default.asp">Home</a></li>
  <li><a href="news.asp">News</a></li>
  <li><a href="contact.asp">Contact</a></li>
  <li><a href="about.asp">About</a></li>
</ul>
```

VERTICAL NAVIGATION BAR:

To build a vertical navigation bar, you can style the <a> elements inside the list, in addition to the code above:

Example

```
li a {
   display: block;
   width: 60px;
}
```

BASIC DROPDOWN:

Create a dropdown box that appears when the user moves the mouse over an element.

Example

```
<style>
.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover .dropdown-content {
  display: block;
}
</style>

<div class="dropdown">
 <span>Mouse over me</span>
 <div class="dropdown-content">
  <p>Hello World!</p>
 </div>
</div>
```

PROGRAM 1: USE OF INTERNAL STYLE SHEET.

```
<html>
<head>
<title>Learning HTML</title>
<style>
```

```
p { color: blue; }
.importantstuff { font-size: 20px; }
#lookhere { font-size: 28px; text-align: center; color: green; }
</style>
</head>
<body>
<p>Woo hoo! I'm learning HTML!</p>
<p class="importantstuff">Woo hoo! I'm learning HTML!</p>
<p class="importantstuff" id="lookhere">Woo hoo! I'm learning HTML!</p>
<p class="importantstuff">Woo hoo! I'm learning HTML!</p>
</body>
</html>
```

OUTPUT:

PROGRAM 2: USE OF EXTERNAL STYLE SHEET.

CSS CODE:

```
body{ background-color: gray;}
p { color: blue; }
h3{ color: white; }
```

HTML CODE:

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="test.css" />
</head>
<body>
```

<h3> A White Header </h3>
** This paragraph has a blue font.**
The background color of this page is gray because
**we changed it with CSS! **
</body>
</html>

OUTPUT: