Chapter 3

3. Cascading Style Sheet (CSS)

3.1. CSS Introduction

What is CSS

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

What does CSS do?

- You can add new looks to your old HTML documents.
- You can completely change the look of your website with only a few changes in CSS code.

Benefits of CSS

These are the three major benefits of CSS:

1) Solves a big problem

Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem. It was a W3C recommendation.

2) Saves a lot of time

CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

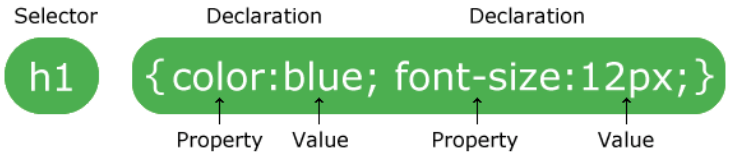
3) Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

3.2. CSS Syntax & how to

CSS Syntax

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



Selector: Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> etc.

Declaration Block: The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

```
color: yellow;
font-size: 11 px;
```

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

Property: A Property is a type of attribute of HTML element. It could be color, border etc.

Value: Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

CSS Selector

CSS selectors are used *to select the content you want to style*. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc. There are several different types of selectors in CSS.

1. CSS Element Selector
2. CSS Id Selector
3. CSS Class Selector
4. CSS Universal Selector
5. CSS Group Selector

1) CSS Element Selector

The element selector selects the HTML element by name.

```
<!DOCTYPE html>
<html>  <head>  <style>
p{      text-align: center;      color: blue;  }
</style>  </head>  <body>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>  <p>And me!</p>
</body>  </html>
```

2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element. It is written with the hash character (#), followed by the id of the element. Let's take an example with the id **"para1".**

```
<!DOCTYPE html>  <html>  <head> <style>
#para1 {  text-align: center;   color: blue; }
</style> </head>  <body>
<p id="para1">Hello Javatpoint.com</p>
<p>This paragraph will not be affected.</p>
</body> </html>
```

3) CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name. A class name should **not be started with a number**.

Let's take an example with a class **"center".**

```
<!DOCTYPE html>
<html>  <head>  <style>
.center {  text-align: center;  color: blue;  }
</style></head>  <body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>  </html>
```

CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector. Let's see an example.

```
<!DOCTYPE html>
<html>  <head>  <style>
p.center {   text-align: center;      color: blue;  }
</style> </head>  <body>
<h1 class="center">This heading is not affected</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body> </html>
```

4) CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
<!DOCTYPE html>
<html> <head>  <style>
* {   color: green;   font-size: 20px;  }
</style>  </head>  <body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>  <p>And me!</p>
</body> </html>
```

5) CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions. Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping. Let's see the CSS code without group selector.

```
 h1 {  text-align: center;  color: blue;  }
h2 {  text-align: center; color: blue;  }
```

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

As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

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

Let's see the full example of CSS group selector.

```
<!DOCTYPE html>  <html>  <head>  <style>
h1, h2, p {  text-align: center;  color: blue;   }
</style>  </head><body>
<h1>Hello Javatpoint.com</h1>
<h2>Hello Javatpoint.com (In smaller font)</h2>
<p>This is a paragraph.</p>  </body>  </html>
```

How to add CSS

CSS is added to HTML pages to format the document according to information in the style sheet. There are three ways to insert CSS in HTML documents.

1. Inline CSS
2. Internal CSS
3. External CSS

Inline CSS

We can apply CSS in a single element by inline CSS technique. The inline CSS is also a method to insert style sheets in HTML document. This method mitigates some advantages of style sheets so it is advised to use this method sparingly. If you want to use inline CSS, you should use the style attribute to the relevant tag.

Syntax:

<htmltag style="cssproperty1:value; cssproperty2:value;"> </htmltag>

Example:

```
<h2 style="color:red;margin-left:40px;">
Inline CSS is applied on this heading.</h2>
<p>This paragraph is not affected.</p>
```

Disadvantages of Inline CSS

- You cannot use quotations within inline CSS. If you use quotations the browser will interpret this as an end of your style value.
- These styles cannot be reused anywhere else.
- These styles are tough to be edited because they are not stored at a single place.
- It is not possible to style pseudo-codes and pseudo-classes with inline CSS.
- Inline CSS does not provide browser cache advantages.

Internal CSS

The internal style sheet is used to add a unique style for a single document. It is defined in <head> section of the HTML page inside the <style> tag.

Example:

```
<!DOCTYPE html>
<html> <head>  <style>
body {      background-color: linen;  }
h1 {      color: red;      margin-left: 80px;  }
</style>  </head>  <body>
<h1>The internal style sheet is applied on this heading.</h1>
<p>This paragraph will not be affected.</p>
</body>  </html>
```

External CSS

The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file. It uses the <link> tag on every pages and the <link> tag should be put inside the head section.

Example:

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

The external style sheet may be written in any text editor but must be saved with a .css extension. This file should not contain HTML elements. Let's take an example of a style sheet file named "**mystyle.css**".

File: mystyle.css

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

Note: You should not use a space between the property value and the unit. For example: It should be margin-left:20px not margin-left:20 px.

CSS Comments

CSS comments are generally written to explain your code. It is very helpful for the users who reads your code so that they can easily understand the code. Comments are ignored by browsers. Comments are single or multiple lines statement and written within /*.....*/ .

```
<!DOCTYPE html>  <html>  <head>  <style>
p {      color: blue;
    /* This is a single-line comment */
    text-align: center;  }
/* This is
a multi-line  comment */
</style>  </head>  <body>
```

```
<p>Hello Javatpoint.com</p>  <p>This statement is styled with CSS.</p>
 <p>CSS comments are ignored by the browsers and not shown in the outp
ut.</p>  </body>  </html>
```

3.3. CSS Specificity

What is Specificity?

If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out. Think of specificity as a score/rank that determines which style declarations are ultimately applied to an element.

The universal selector (*) has low specificity, while ID selectors are highly specific!

Note: Specificity is a common reason why your CSS-rules don't apply to some elements, although you think they should.

Specificity Hierarchy

Every selector has its place in the specificity hierarchy. There are four categories which define the specificity level of a selector:

Inline styles - An inline style is attached directly to the element to be styled.

Example: <h1 style="color: #ffffff;">.

IDs - An ID is a unique identifier for the page elements, such as #navbar.

Classes, attributes and pseudo-classes - This category includes .classes, [attributes] and pseudo-classes such as :hover, :focus etc.

Elements and pseudo-elements - This category includes element names and pseudo-elements, such as h1, div, :before and :after.

How to Calculate Specificity?

Memorize how to calculate specificity!

Start at 0, add 1000 for style attribute, add 100 for each ID, add 10 for each attribute, class or pseudo-class, add 1 for each element name or pseudo-element. Consider these three code fragments:

Example

```
A:                                                              h1
B:                              #content                        h1
C: <div id="content"><h1 style="color: #ffffff">Heading</h1></div>
```

The specificity of A is 1 (one element)

The specificity of B is 101 (one ID reference and one element)

The specificity of C is 1000 (inline styling)

Since 1 < 101 < 1000, the third rule (C) has a greater level of specificity, and therefore will be applied.

Specificity Rules

Equal specificity: the latest rule counts - If the same rule is written twice into the external style sheet, then the lower rule in the style sheet is closer to the element to be styled, and therefore will be applied:

Example:

```
h1 {background-color: yellow;}
h1 {background-color: red;}
```

the latter rule is always applied.

ID selectors have a higher specificity than attribute selectors - Look at the following three code lines:

Example:

```
div#a {background-color: green;}
#a {background-color: yellow;}
div[id=a] {background-color: blue;}
```

the first rule is more specific than the other two, and will be applied.

Contextual selectors are more specific than a single element selector - The embedded style sheet is closer to the element to be styled. So in the following situation

Example

From external CSS file: `#content h1 {background-color: red;}`

In HTML file: `<style> #content h1 {background-color: yellow; }</style>`

the latter rule will be applied.

A class selector beats any number of element selectors - a class selector such as .intro beats h1, p, div, etc:

Example

```
.intro {background-color: yellow;}
h1 {background-color: red;}
```

The universal selector and inherited values have a specificity of 0 - *, body * and similar have a zero specificity. Inherited values also have a specificity of 0.

3.4. CSS Colors, Backgrounds, Borders, Margins, Padding & Height/Width

CSS Colors

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

Color Names

In HTML, a color can be specified by using a color name like Tomato, Orange, DodgerBlue, MediumSeaGreen, Gray, SlateBlue, Violet, LightGray etc.

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<!– background color -->
<p style="color:MediumSeaGreen;"></p> <!-- Text Color -->
<h1 style="border:2px solid Violet;">Hello World</h1>
<!-- Border Color -->
```

Color Values

In HTML, colors can also be specified using Color names, RGB values, HEX values etc:

Example: `color:tomato; or color:rgb(255, 99, 71); or color:#ff6347;`

RGB Value

In HTML, a color can be specified as an RGB value, using this formula: **rgb(*red, green, blue*)**

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.To display the color black, all color parameters must be set to 0, like this: rgb(0, 0, 0). To display the color white, all color parameters must be set to 255, like this: rgb(255, 255, 255).

HEX Value

In HTML, a color can be specified using a hexadecimal value in the form: ***#rrggbb***

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

CSS Background

CSS background property is used to define the background effects on element. There are different CSS background properties that affect the HTML elements:

Property	Description
background	Sets all the background properties in one declaration
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page
background-clip	Specifies the painting area of the background
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-origin	Specifies where the background image(s) is/are positioned
background-position	Sets the starting position of a background image
background-repeat	Sets how a background image will be repeated
background-size	Specifies the size of the background image(s)

Example:

```
<!DOCTYPE html>
<html><head> <style>
h2,p{
```

```
background-color:#b0d4de; /*color name/rgb value is allowed too*/
background-image: url("image1.jpg");
background-repeat: no-repeat; /*other values: repeat-x/repeat-y*/
background-attachment: fixed;
background-position: center; /*other values:left/right*/    }
</style> </head>  <body>
 <h2>My first CSS page.</h2>
 <p>This is an example of CSS background properties</p>
</body>  </html>
```

CSS Border

The CSS border is a shorthand property used to set the border on an element. The CSS border properties are use to specify the style, color and size of the border of an element. The CSS border properties are: border-style, border-color, border-width, border-radius etc.

Property	**Description**
border	Sets all the border properties in one declaration
border-bottom	Sets all the bottom border properties in one declaration
border-bottom-color	Sets the color of the bottom border
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-color	Sets the color of the four borders
border-left	Sets all the left border properties in one declaration
border-left-color	Sets the color of the left border
border-left-style	Sets the style of the left border
border-left-width	Sets the width of the left border
border-radius	Sets all the four border-*-radius properties for rounded corners
border-right	Sets all the right border properties in one declaration
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-style	Sets the style of the four borders
border-top	Sets all the top border properties in one declaration
border-top-color	Sets the color of the top border
border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border

border-width	Sets the width of the four borders

Example:

```
<html><head><style>
 div{
      border-style:solid;
      border-width:2px;
      border-color:blue;
      border-radius:5px;
}
</style></head><body>
<div><header>DMU Website</header>
<nav><a href="#">Home</a></nav>
<section><h1>News</h1><p>The First news…</p></section>
<footer>Developed by:</footer>
</div></body></html>
```

The `border` property is a shorthand property for the following individual border properties: `border-width, border-style` (required) and `border-color`

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

CSS Margin

CSS Margin property is used to define the space around elements. It is completely transparent and doesn't have any background color. It clears an area around the element. Top, bottom, left and right margin can be changed independently using separate properties. You can also change all properties at once by using shorthand margin property.

CSS Margin Properties

Property	**Description**
Margin	This property is used to set all the properties in one declaration.
margin-left	it is used to set left margin of an element.
margin-right	It is used to set right margin of an element.
margin-top	It is used to set top margin of an element.
margin-bottom	It is used to set bottom margin of an element.

CSS Margin Values

These are some possible values for margin property.

Value	**Description**
auto	This is used to let the browser calculate a margin.
length	It is used to specify a margin pt, px, cm, etc. its default value is 0px.
%	It is used to define a margin in percent of the width of containing element.

inherit	It is used to inherit margin from parent element.

Note: You can also use negative values to overlap content.

CSS margin Example

You can define different margin for different sides for an element.

```
<!DOCTYPE html>  <html>  <head>  <style>
p {  background-color: pink;  }
p.ex {  margin-top: 50px;  margin-bottom: 50px;
        margin-right: 100px;   margin-left: 100px;
}
</style>  </head>  <body>
<p>This paragraph is not displayed with specified margin. </p>
<p class="ex">This paragraph is displayed with specified margin.</p>
</body> </html>
```

Margin: Shorthand Property

CSS shorthand property is used to shorten the code. It specifies all the margin properties in one property.

There are four types to specify the margin property. You can use one of them.

```
margin: 50px 100px 150px 200px; /* defines top, right, bottom and left margins respectively*/
margin: 50px 100px 150px; /* defines top,right and left margins respectively*/
margin: 50px 100px; /* defines 50px for top and bottom, 100px for left and right*/
margin 50px; /* defines equal margin for all */
```

CSS Padding

CSS Padding property is used *to define the space between the element content and the element border*.

It is different from CSS margin in the way that CSS margin defines the space around elements. CSS padding is affected by the background colors. It clears an area around the content.

Top, bottom, left and right padding can be changed independently using separate properties. You can also change all properties at once by using shorthand padding property.

CSS Padding Properties

Property	Description
Padding	It is used to set all the padding properties in one declaration.
padding-left	It is used to set left padding of an element.
padding-right	It is used to set right padding of an element.
padding-top	It is used to set top padding of an element.
padding-bottom	It is used to set bottom padding of an element.

CSS Padding Values

Value	Description
length	It is used to define fixed padding in pt, px, em etc.
%	It defines padding in % of containing element.

Example

```
<!DOCTYPE html>  <html> <head> <style>
p {  background-color: pink;  }
p.padding {  padding-top: 50px;     padding-right: 100px;
             padding-bottom: 150px;   padding-left: 200px;  }
</style> </head>  <body>
<p>This is a paragraph with no specified padding.</p>
<p class="padding">This is a paragraph with specified paddings.</p>
</body>  </html>
```

CSS 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; }
```

Example

```
div {height: 100px; width: 500px;    background-color: powderblue; }
```

Note: The `height` and `width` properties do not include padding, borders, or margins; they set the height/width of the area inside the padding, border, and margin of the element!

Setting max-width

The `max-width` property is used to set the maximum width of an element.

The `max-width` can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default i.e. there is no maximum width).

The problem with the `<div>` above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows.

Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

Note: The value of the `max-width` property overrides `width`.

The following example shows a `<div>` element with a height of 100 pixels and a max-width of 500 pixels:

Example

```
div { max-width: 500px; height: 100px; background-color: powderblue; }
```

All CSS Dimension Properties

Property	Description
height	Sets the height of an element
max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element
min-height	Sets the minimum height of an element
min-width	Sets the minimum width of an element
width	Sets the width of an element

3.5. CSS text, Fonts, Links, Lists, Tables, Image & forms

CSS Text

CSS has different properties to change the appearance of a text. It consists of CSS properties for text color, alignment, decoration, transformation, indentation, spacing, shadow etc.

Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a text
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the decoration added to text
text-indent	Specifies the indentation of the first line in a text-block
text-shadow	Specifies the shadow effect added to text
text-transform	Controls the capitalization of text
text-overflow	Specifies how overflowed content that is not displayed should be signaled to the user
unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
vertical-align	Sets the vertical alignment of an element
white-space	Specifies how white-space inside an element is handled
word-spacing	Increases or decreases the space between words in a text

Example:

```
<html><head><style>
```

```
article{line-height:20px; border:1px solid blue; border-radius:3px;}
h1{color:blue; text-align:center; text-decoration:underline;
   text-transform:capitalize; text-shadow:3px 2px yellow;}
p{color:#009933;text-align:justify;word-spacing:2px;text-indent:3px;}
</style></head><body><header>DMU Website</header>
<nav><a href="#">Home</a></nav>
<section><article>
     <h1>What is new?</h1><p> To Debre Markos University …..</p>
</article></section>
<footer>Developed by:</footer></body></html>
```

CSS Fonts

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

CSS Font Families

In CSS, there are two types of font family names:

- ✓ generic family - a group of font families with a similar look (like "Serif" or "Monospace")
- ✓ font family - a specific font family (like "Times New Roman" or "Arial")

Generic family	**Font family**	**Description**
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Note: On computer screens, sans-serif fonts are considered easier to read than serif fonts.

In addition to font-family, CSS has different font properties. These are:

Property	**Description**
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text. Defined in em or px. 16px=1em (default size for paragraph or normal text)
font-style	Specifies the font style for text. The values of this properties are normal, italic and oblique
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font. Values include bold or normal

Example:

```
h1{font-size:2em; font-weight: bold; }
```

```
p{font-size:12px; font-family: "Times New Roman", Times, serif;}
```

CSS Links

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

Example

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

In addition, links can be styled differently depending on what **state** they are in.

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 moves over it
- ✓ a:active - a link the moment it is clicked

Example

```
/*                  unvisited                  link                  */
a:link             {              color:              red;             }
/*                   visited                   link                  */
a:visited           {             color:            green;            }
/*              mouse              over              link              */
a:hover                {  color:                hotpink;              }
/*                  selected                   link                  */
a:active {  color: blue; }
```

When setting the style for several link states, there are some order rules:

- ✓ a:hover MUST come after a:link and a:visited
- ✓ a:active MUST come after a:hover

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

Example

```
a:link          {           text-decoration:          none;          }
a:visited         {           text-decoration:          none;          }
a:hover           {  text-decoration:           underline;          }
a:active {  text-decoration: underline; }
```

Background Color

The background-color property can be used to specify a background color for links:

Example

```
a:link            {  background-color:            yellow;            }
a:visited         {          background-color:         cyan;          }
```

```
a:hover { background-color: lightgreen; }
a:active { background-color: hotpink; }
```

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

In HTML, there are two main types of lists:

- ✓ unordered lists (<ul>) - the list items are marked with bullets
- ✓ ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

- ✓ Set different list item markers for ordered lists
- ✓ Set different list item markers for unordered lists
- ✓ Set an image as the list item marker

Property	Description
list-style	Sets all the properties for a list in one declaration
list-style-image	Specifies an image as the list-item marker
list-style-position	Specifies the position of the list-item markers (bullet points)
list-style-type	Specifies the type of list-item marker

Example:

CSS Tables

Table Borders

To specify table borders in CSS, use the border property. The example below specifies a black border for <table>, <th>, and <td> elements:

Example

```
table, th, td { border: 1px solid black; }
```

Notice that the table in the example above has double borders. This is because both the table and the <th> and <td> elements have separate borders.

Collapse Table Borders

The border-collapse property sets whether the table borders should be collapsed into a single border:

Example

```
table { border-collapse: collapse; }
table, th, td { border: 1px solid black; }
```

If you only want a border around the table, only specify the border property for <table>:

Example

```
table { border: 1px solid black;}
```

Table Width and Height

Width and height of a table are defined by the width and height properties. The example below sets the width of the table to 100%, and the height of the <th> elements to 50px:

Example

```
table { width: 100%; }
th { height: 50px; }
```

Horizontal Alignment

The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>. By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned. The following example left-aligns the text in <th> elements:

Example: `th { text-align: left; }`

Vertical Alignment

The vertical-align property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>. By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements). The following example sets the vertical text alignment to bottom for <td> elements:

Example

```
td { height: 50px; vertical-align: bottom;}
```

Table Padding

To control the space between the border and the content in a table, use the padding property on <td> and <th> elements:

Example

```
th, td { padding: 15px; text-align: left; }
```

Horizontal Dividers

Add the border-bottom property to <th> and <td> for horizontal dividers:

Example

```
th, td {  border-bottom: 1px solid #ddd; }
```

Hoverable Table

Use the :hover selector on <tr> to highlight table rows on mouse over:

Example

```
tr:hover {background-color: #f5f5f5;}
```

Striped Tables

For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows:

Example

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

Table Color

The example below specifies the background color and text color of <th> elements:

Example

```
th {  background-color: #4CAF50;   color: white; }
```

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content. Add a container element (like) with overflow-x:auto around the <table> element to make it responsive:

Example

```
<div style="overflow-x:auto;"><table>...table content...</table></div>
```

Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

CSS Forms

Styling Input Fields

Use the width property to determine the width of the input field:

Example

```
input {  width: 100%; }
```

The example above applies to all <input> elements. If you only want to style a specific input type, you can use attribute selectors:

- ✓ input[type=text] - will only select text fields
- ✓ input[type=password] - will only select password fields
- ✓ input[type=number] - will only select number fields etc..

Padded Inputs

Use the padding property to add space inside the text field.

Tip: When you have many inputs after each other, you might also want to add some margin, to add more space outside of them:

Example

```
input[type=text] {
width:100%; padding:12px 20px; margin: 8px 0; box-sizing: border-box;}
```

Note that we have set the box-sizing property to border-box. This makes sure that the padding and eventually borders are included in the total width and height of the elements.

Bordered Inputs

Use the border property to change the border size and color, and use the border-radius property to add rounded corners:

Example

```
input[type=text] {  border: 2px solid red;  border-radius: 4px; }
```

If you only want a bottom border, use the border-bottom property:

Example

```
input[type=text] {  border: none;  border-bottom: 2px solid red; }
```

Colored Inputs

Use the background-color property to add a background color to the input, and the color property to change the text color:

Example

```
input[type=text] {  background-color: #3CBC8D;  color: white; }
```

Focused Inputs

By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input. Use the :focus selector to do something with the input field when it gets focus:

Example

```
input[type=text]:focus {  background-color: lightblue; }
```

Example

```
input[type=text]:focus {  border: 3px solid #555; }
```

Input with icon/image

If you want an icon inside the input, use the background-image property and position it with the background-position property. Also notice that we add a large left padding to reserve the space of the icon:

Example

```
input[type=text]                                                    {
                        background-color:                          white;
                background-image:              url('searchicon.png');
              background-position:             10px              10px;
                     background-repeat:                    no-repeat;
```

```
                                padding-left:                                40px;
}
```

Animated Search Input

In this example we use the CSS transition property to animate the width of the search input when it gets focus.

Example

```
input[type=text]  {   -webkit-transition:  width  0.4s  ease-in-out;
transition:          width          0.4s          ease-in-out;          }
input[type=text]:focus {  width: 100%; }
```

Styling Textareas

Tip: Use the resize property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):

Example

```
textarea  {     width:  100%;   height:  150px;   padding:  12px  20px;
  box-sizing: border-box; border: 2px solid #ccc; border-radius: 4px;
         background-color:        #f8f8f8;        resize:        none;
}
```

Styling Select Menus

Example

```
select                                                                  {
     width:    100%;     padding:    16px    20px;     border:    none;
         border-radius:         4px;  background-color:         #f1f1f1;
}
```

Styling Input Buttons

Example

```
input[type=button],     input[type=submit],     input[type=reset]     {
    background-color:   #4CAF50;    border:   none;    color:   white;
       padding:       16px       32px;        text-decoration:       none;
       margin:       4px       2px;        cursor:       pointer;       }
```

/* Tip: use **width: 100%** for full-width buttons */

CSS Images

Rounded Images

Use the `border-radius` property to create rounded images:

Example: `img {  border-radius: 8px;  }`

Circled Image

Example: `img {  border-radius: 50%;  }`

Thumbnail Images

Use the `border` property to create thumbnail images.

Example:

```
img{border:1px solid #ddd; border-radius:4px; padding:5px;width:50px;}
```

Responsive Images

Responsive images will automatically adjust to fit the size of the screen. Resize the browser window to see the effect. If you want an image to scale down if it has to, but never scale up to be larger than its original size, add the following:

Example

```
img {  max-width: 100%;  height: auto; }
```

Center an Image

To center an image, set left and right margin to `auto` and make it into a `block` element:

Example

```
img {display:block; margin-left:auto; margin-right:auto; width:50%; }
```

Image gallery

CSS can be used to create an image gallery. The following image gallery is created with CSS:

Example:

```
<html>                          <head>                          <style>
div.gallery{margin:5px; border:1px solid #ccc; float:left;
width:180px;                                                    }
div.gallery:hover       {  border:1px       solid       #777;       }
div.gallery          img          { width:100%; height:auto;          }
div.desc       {        padding:15px;       text-align:center;       }
</style></head><body>
<div  class="gallery">  <a  target="_blank"  href="fjords.jpg">  <img
src="5terre.jpg"  alt="Cinque  Terre"  width="300"  height="200"></a>
   <div  class="desc">Add  a  description  of  the  image  here</div>
</div>
<div     class="gallery"><a     target="_blank"     href="forest.jpg">
    <img  src="forest.jpg"  alt="Forest"  width="300"  height="200"></a>
   <div  class="desc">Add  a  description  of  the  image  here</div>
</div>
<div    class="gallery">    <a    target="_blank"    href="lights.jpg">
 <img src="lights.jpg" alt="Northern Lights" width="300" height="200">
                                                                </a>
  <div class="desc">Add a description of the image here</div></div>
<div class="gallery"><a target="_blank" href="mountains.jpg">
<img src="mountains.jpg" alt="Mountains" width="300" height="200"></a>
   <div  class="desc">Add  a  description  of  the  image  here</div>
</div> </body> </html>
```

3.6. CSS box model, outline, display, max width, position, float, Inline block, navigation bar and dropdowns

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 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;}
```

Width and Height of an Element

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Important: When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add padding, borders and margins.

Assume we want to style a element to have a total width of 350px:

Example:`div{width:320px; padding:10px;border:5px solid gray;  margin:0;}`

The total width of an element should be calculated like this:

Total element width=width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should be calculated like this:

Total element height=height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

CSS Outline

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out". CSS has the following outline properties: `outline-style`, `outline-color`, `outline-width`, `outline-offset` and `outline`.

- The `outline-style` property specifies the style of the outline. Values for `outline-style` are similar with border-style.
- The `outline-color` property is used to set the color of the outline.
- The `outline-width` property specifies the width of the outline, and can have one of the following values: thin (typically 1px), medium (typically 3px) and thick (typically 5px). A specific size (in px, pt, cm, em, etc)
- The `outline` property is a shorthand property for setting the following individual outline properties: `outline-width`, `outline-style` (required), `outline-color`
- The `outline-offset` property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

Note: Outline differs from borders! Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline.

Display

The `display` property is the most important CSS property for controlling layout. 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`. `display: none;` is commonly used with JavaScript to hide and show elements without deleting and recreating them. Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards. A common example is making inline `<li>` elements for horizontal menus:

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

Note: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with `display: block;` is not allowed to have other block elements inside it.

Max width

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. This element has a width of 500px, and margin set to auto.

Note: The problem with the `<div>` above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices. This element has a max-width of 500px, and margin set to auto.

Tip: Resize the browser window to less than 500px wide, to see the difference between the two divs!

Here is an example of the two divs above:

Position

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

There are five different position values: `static, relative, fixed, absolute & sticky.`

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default. Static positioned elements are not affected by the top, bottom, left, and right properties. An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page. This element has `position: static;`

Here is the CSS that is used:

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

position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element. This element has `position: relative;`

Here is the CSS that is used:

Example

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

position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element. A fixed element does not leave a gap in the page where it would normally have been located. Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example: `div.fixed {position:fixed; bottom:0;right:0; width:300px; border: 3px solid #73AD21; }`

This element has `position: fixed;`

position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed). However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except `static`.

Here is a simple example:

```
div.relative { position:relative; width:400px; height:200px;
border: 3px solid #73AD21; }
div.absolute { position: absolute; top: 80px; right: 0;
width: 200px; height: 100px; border: 3px solid #73AD21; }
```

position: sticky;

An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Note: Internet Explorer, Edge 15 and earlier versions do not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of `top`, `right`, `bottom` or `left` for sticky positioning to work.

In this example, the sticky element sticks to the top of the page (`top: 0`), when you reach its scroll position.

Example

```
div.sticky {
    position: -webkit-sticky; /* Safari */
    position: sticky;
    top: 0;
    background-color: green;
    border: 2px solid #4CAF50;
}
```

The float Property

The CSS `float` property specifies how an element should float. The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The `float` property can have one of the following values:

- left - The element floats to the left of its container. Example: `img { float: left; }`
- right- The element floats to the right of its container. Example: `img { float: right; }`
- none - The element does not float (will be displayed just where it occurs in the text). This is default. Example: `img { float: none; }`
- inherit - The element inherits the float value of its parent

In its simplest use, the `float` property can be used to wrap text around images.

The clear Property

The `clear` property specifies what elements can float beside the cleared element and on which side. The `clear` property can have one of the following values:

- none - Allows floating elements on both sides. This is default
- left - No floating elements allowed on the left side. Example: `div { clear: left; }`
- right- No floating elements allowed on the right side
- both - No floating elements allowed on either the left or the right side
- inherit - The element inherits the clear value of its parent

The most common way to use the `clear` property is after you have used a `float` property on an element. When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

Navigation Menu

Use `float` with a list of hyperlinks to create a horizontal menu:

Example

- Home
- News
- Contact
- About

Navigation bar

Having easy-to-use navigation is important for any web site. With CSS you can transform boring HTML menus into good-looking navigation bars.

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>
```

Now let's remove the bullets and the margins and padding from the list:

Example

```
ul { list-style-type: none; margin: 0; padding: 0; }
```

- `list-style-type: none;` - Removes the bullets. A navigation bar does not need list markers
- Set `margin: 0;` and `padding: 0;` to remove browser default settings

The code in the example above is the standard code used in both vertical, and horizontal navigation bars.

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; }`

- `display: block;` - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width (and padding, margin, height, etc. if you want)
- `width: 60px;` - Block elements take up the full width available by default. We want to specify a 60 pixels width

You can also set the width of <ul>, and remove the width of <a>, as they will take up the full width available when displayed as block elements. This will produce the same result as our previous example:

Example

```
ul { list-style-type:none; margin:0; padding:0; width:60px; }
li a { display: block;}
```

Vertical Navigation Bar Examples

Create a basic vertical navigation bar with a gray background color and change the background color of the links when the user moves the mouse over them.

Example

```
ul { list-style-type:none; margin:0; padding:0; width:200px;
                                background-color: #f1f1f1; }
li a{display:block; color:#000;padding:8px 16px;text-decoration:none;}
/* Change the link color on hover */
li a:hover {background-color:#555; color:white; }
```

Center Links & Add Borders

Add `text-align:center` to <li> or <a> to center the links. Add the `border` property to <ul> add a border around the navbar. If you also want borders inside the navbar, add a `border-bottom` to all <li> elements, except for the last one:

Example

```
ul { border: 1px solid #555; }
li { text-align: center; border-bottom: 1px solid #555; }
li:last-child { border-bottom: none; }
```

Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.

Inline List Items

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code above:

Example: `li { display: inline; }`

- `display: inline;` - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Floating List Items

Another way of creating a horizontal navigation bar is to float the <li> elements, and specify a layout for the navigation links:

Example

```
li { float: left; }
a { display: block; padding: 8px; background-color: #dddddd;}
```

Example explained:

- `float: left;` - use float to get block elements to slide next to each other

- `display: block;` - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify padding (and height, width, margins, etc. if you want)
- `padding: 8px;` - Since block elements take up the full width available, they cannot float next to each other. Therefore, specify some padding to make them look good
- `background-color: #dddddd;` - Add a gray background-color to each a element

Tip: Add the background-color to <ul> instead of each <a> element if you want a full-width background color:

Example: `ul { background-color: #dddddd; }`

Horizontal Navigation Bar Examples

Create a basic horizontal navigation bar with a dark background color and change the background color of the links when the user moves the mouse over them:

Example

```
ul {list-style-type:none; margin:0; padding:0;
overflow: hidden; background-color: #333; }
li { float:left; }
li a { display:block; color:white; text-align:center;
padding: 14px 16px; text-decoration: none; }
/* Change the link color to #111 (black) on hover */
li a:hover { background-color: #111; }
```

Border Dividers

Add the `border-right` property to <li> to create link dividers:

Example

```
/* Add a gray right border to all list items, except the last item
(last-child) */
li { border-right: 1px solid #bbb; }
li:last-child { border-right: none; }
```

Dropdowns

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>
```

Example Explained

HTML) Use any element to open the dropdown content, e.g. a , or a <button> element. Use a container element (like) to create the dropdown content and add whatever you want inside of it. Wrap a element around the elements to position the dropdown content correctly with CSS.

CSS) The `.dropdown` class uses `position:relative`, which is needed when we want the dropdown content to be placed right below the dropdown button (using `position:absolute`). The `.dropdown-content` class holds the actual dropdown content. It is hidden by default, and will be displayed on hover (see below).

Note the `min-width` is set to 160px. Feel free to change this.

Tip: If you want the width of the dropdown content to be as wide as the dropdown button, set the `width` to 100% (and `overflow:auto` to enable scroll on small screens). Instead of using a border, we have used the CSS `box-shadow` property to make the dropdown menu look like a "card".

The `:hover` selector is used to show the dropdown menu when the user moves the mouse over the dropdown button.

Dropdown Menu

Create a dropdown menu that allows the user to choose an option from a list:

This example is similar to the previous one, except that we add links inside the dropdown box and style them to fit a styled dropdown button:

Example

```
<style>      /*      Style      The      Dropdown      Button      */
.dropbtn {  background-color: #4CAF50;  color: white;  padding: 16px;
    font-size:   16px;    border:   none;    cursor:   pointer;   }
/* The container <div> - needed to position the dropdown content */
.dropdown   {    position:   relative;    display:   inline-block;   }
/*      Dropdown      Content      (Hidden      by      Default)      */
.dropdown-content   {    display:   none;    position:   absolute;
        background-color:       #f9f9f9;        min-width:       160px;
   box-shadow:  0px  8px  16px  0px  rgba(0,0,0,0.2);   z-index:  1;  }
/*        Links        inside        the        dropdown        */
.dropdown-content a{color:black; padding:12px 16px;
text-decoration:        none;         display:        block;        }

/*    Change    color    of    dropdown    links    on    hover    */
.dropdown-content       a:hover       {background-color:       #f1f1f1}
```

```
/* Show the dropdown menu on hover */
.dropdown:hover .dropdown-content { display: block; }
/* Change the background color of the dropdown button when the
dropdown content is shown */
.dropdown:hover .dropbtn { background-color: #3e8e41; }
</style>
<div class="dropdown"> <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content"> <a href="#">Link 1</a>
   <a href="#">Link 2</a><a href="#">Link 3</a> </div>
</div>
```

Right-aligned Dropdown Content

If you want the dropdown menu to go from right to left, instead of left to right, add `right: 0;`

Example: `.dropdown-content { right: 0; }`

3.7. CSS Responsive

Responsive web design makes your web page look good on all devices. Responsive web design uses only HTML and CSS. Responsive web design is not a program or a JavaScript. Web pages can be viewed using many different devices: desktops, tablets, and phones. Your web page should look good, and be easy to use, regardless of the device. Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:

Desktop **Tablet** **Phone**

It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

Reading Assignment

Refer the examples for dropdown menu, responsive design and other advanced CSS properties at www.w3schools.com.