IT664 Chapter 2

Cascading Color Sheets (CSS3)

1. What is a CSS?

HTML provides structure to an HTML document. HTML code does not provide the browser any presentation information for the HTML elements. Cascading Style Sheets (CSS) is a mechanism to associate presentation information with an HTML document. CSS controls the look and format of HTML elements. In other words, CSS makes your Web pages beautiful!

A CSS style sheet is a text file that contains declarations which specify how elements in an HTML document is to be displayed. A CSS file is normally saved with the extension .css.

The first version of CSS, CSS1, was introduced by W3C in 1996. The next version, CSS2 in 1998. The current version is CSS level 3.
(See http://www.w3.org/Style/CSS/current-work.en.html)

Even though CSS3 is still under development, latest browsers support most of the features of CSS3. I use the latest Chrome in my examples.

A CSS declaration consists of a property name and a value associated with the property, separated by a colon:

```
propertyName : value;
```

Example:

Here is a CSS declaration:

```
color:red;
```

Here color is a CSS property and red is a value. This CSS declaration is specifying that the text color of the matching HTML element be red. We will see soon how to match HTML elements with a CSS declaration.

Example:

Here is another CSS declaration:

```
font-size :15px;
```

This CSS declaration is specifying that the font size of the text of the element be 15 pixels.

Braces are used to group a set of CSS declarations:

```
{
  propertyName : value;
  propertyName : value;
  propertyname : value;
}
```

Such a group of properties is called a CSS rule.

We then match a CSS rule with a set of HTML elements. This set of matching elements in the document is called a selector.

Syntax for matching HTML elements:

```
selector
{
  propertyName : value;
  propertyName : value;
  propertyname : value;
}
```

The browser will apply the specified rules to all the elements given by the *selector* before rendering the elements on the screen.

EXAMPLE:

```
h1
{
  color : blue;
  font-size :15px;
}
```

Here h1 is the selector. The two properties, color (blue) and font-size (15px) are applied to the contents of all **<h1>** tags in the HTML document. In other words, the

browser will render contents of all **<h1>** elements in color blue and font-size 15 pixels.

In the rest of this chapter, we will see some of the various ways of specifying selectors and some of the hundreds of CSS properties and values.

2. Various ways of specifying a selector

There are several ways you can specify selectors (that is, patterns specifying elements to apply the CSS rules).

There are more than 30 ways of specifying a selector. We will discuss only commonly used selectors.

➢ A selector can be an HTML tag

A selector can be a tag name. The rules will be applied to the contents of the indicated tags.

EXAMPLE:

```
h1
{
  font-size: 15px;
  color:blue
}
```

This is specifying that the text within **<h1> … </h1>** is to be of displayed with font size of 15 pixels, in blue color. The rules will applied to contents of all <h1>elements in the HTML document.

➢ A selector can be several tags

The selector can be a list of tags names, separated by commas.

EXAMPLE:

```
h1, div
{

}
```

The rules will be applied to the contents of all **<h1>** and ** ** tags. The comma is required (don't forget it). It means something else without comma.

- **A selector can be a class attribute value with a preceding period (.)**

If you like to apply certain CSS rules to several selected tags, then use the **class** attribute in these tags and assign the same class value in all these tags. The selector *.value* selects these tags with this value for the class attribute.

EXAMPLE:

Consider the following HTML tags with a **class** attribute and the same value:

```
<h1 class="sample">
<div class="sample">
```

Now, the syntax for applying certain CSS rules to these <h1> and elements is,

```
.sample
{

}
```

This CSS rule will be applied to the contents of all the tags, which have the **class** attribute value *sample*. The rules will not be applied any tag without **class**="*sample*" attribute.

NOTE:

Notice the period before the class value in the selector.

- **A selector can be an id attribute value with a preceding #**

In a selector, instead of using the value of a class attribute, you can use the value of an **id** attribute. In this case, use # instead of a period (.) as the first character in the selector.

Remember that the **id** attribute values are unique in an HTML document. No two tags can have the same value for the **id** attribute. In contrast, several tags in an HTML document can have the same value for the **class** attribute.

EXAMPLE:

```
<h1 id="id1">

#id1
{

}
```

This CSS rule applies to the tag with **id** value, *id1*.

NOTE:

Remember that when you start a selector with a period, it is referring to a **class** attribute value and when it starts with a #, it is referring to an **id** attribute value.

- **A selector can be a tag within another tag**

The syntax,

```
tag1 tag2
{

}
```

Implies that the CSS rule is to be applied to *tag2* within *tag1*.

EXAMPLE:

Consider HTML code,

```
<h1>An example</h1>
<div> Here is an example of a big city: <h1>New York.</h1>  Have you visited NYC?</div>
```

The following CSS rule,

```
div  h1
{
    color:blue;
    font-size:15px;
}
```

Applies to the contents of <h1> appearing only within .

Note:

Remember,

- Selector *tag1 tag2* selects the element *tag2* within *tag1*.
- Selector *tag1, tag2* selects both tags, *tag1* and *tag2*.

- **A selector to select all tags**

The * selector selects ALL tags in the HTML document.

This is generally used to set basic properties to the entire document. Examples of such properties are margin and border-color – you may want to set these for the entire document.

Note:

There are several more selectors for sophisticated situations. Let us skip them for now. Read online manuals for them.

Note:

- There must be a semicolon at the end of each CSS declaration. There is no need to place a semicolon after the last declaration.
- Several CSS declarations can be placed on the same line (separated by semicolons).
- Comments are placed in a CSS file using the syntax: /* ... */.
- A CSS rule always starts on a new line.

Note:

- **You can include multiple sets of rules for the same selector:**

  For a selector you can provide several sets of CSS declarations:

selector{...}
sameselector{ ...}
sameselector{...}

- **Inheritance of values**

Values of certain properties declared for an element are inherited by the child elements. Most properties are inherited.

3. Associating CSS styles with an HTML document

There are several ways of referencing a set of CSS rules with HTML documents and elements.

- **External CSS style sheet**

You can place all your CSS declarations in a text file and reference the file name with an HTML document. Such a file is called CSS style sheet and is saved with the extension .css.

A CSS style sheet file is associated with an HTML document using the tag **<link>** in the HTML document.

Syntax for <link> tag

<link rel="stylesheet" href="*CSS file name*" >

Where *CSS file name* is the name of the CSS style sheet file including the path (if it is not in the current directory), or the URL if it is on a server on the Internet.

The **<link>** tag goes between **<head>…</head>.**

Note:

- Once you create a CSS style sheet file, you can use the same file in several HTML documents (using <link> in each).
- The link tag has another attribute: type="text/css". Using this is optional. In HTML5 it is not required.

- **Internal style sheet**

If you have CSS styles you like to use in just the current HTML document, declare the styles within **<style>…</style>.**

The **<style>…</style>** go within **<head>…</head>.**

Syntax:

```
<head>
<style>
selector
{
  propertyName: value;
  propertyName: value;
  propertyName: value
}
</style>
</head>
<body>
...
...
...
</body>
```

- **Inline styles**

If you like to define styles just for one tag, then define the styles in the tag itself.

Syntax:

```
<tagName  style="CSS rule" >...</tagName>
```

The style rule applies just for this tag.

EXAMPLE:

```
<h2 style="color:red; font-size:20px">New York City</h2>
```

Note:

This method of specifying CSS styles is not encouraged by experts.

PRACTICE EXAMPLE 1

PracticeExample1.css

```
h2{color:red; font-size:20px}
h3{color:cyan; font-size:15px}
div{color:magenta; font-size:15px}
```

Save this in a file, PracticeExample1.css.

PracticeExample1.HTML

```
<!DOCTYPE html>
<html>
  <head>
   <link rel="stylesheet"  href=" PracticeExample1.css">
  </head>
  <body >
   <h2>New York City</h2>
   <div>
        New York is the most populous city in the United States
        and the center of the New York Metropolitan Area.
    </div>
    <h3>URL: www.nyc.gov</h3>
 </body>
</html>
```

Chrome output:

New York City

New York is the most populous city in the United States and the center of the New York Metropolitan Area.

URL: www.nyc.gov

Same example with internal style sheet

```
!DOCTYPE html>
<html>
  <head>
   <style>
      h2{color:red; font-size:20px}
      h3{color:cyan; font-size:15px}
      div{color:magenta; font-size:15px}
   </style>
  </head>
  <body >
```

```
    <h2>New York City</h2>
    <div>
        New York is the most populous city in the United States
        and the center of the New York Metropolitan Area.
     </div>
     <h3>URL: www.nyc.gov</h3>
  </body>
</html>
```

Same example with inline styles

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body >
    <h2 style="color:red; font-size:20px">New York City</h2>
    <div style="color:magenta; font-size:15px">
        New York is the most populous city in the United States
        and the center of the New York Metropolitan Area.
     </div>
     <h3 style="color:cyan; font-size:15px">URL: www.nyc.gov</h3>
  </body>
</html>
```

4. Block and inline elements

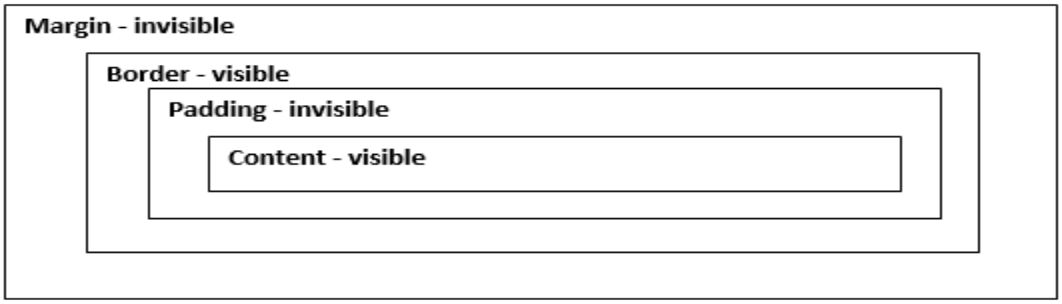
Here is a concept, which is not directly related to CSS, but comes into play in several places related to CSS. In general, an element which results in starting the content on a new line is called a block-level element. Elements which do not produce a line break are called inline elements. For example, in HTML, the tag <h1>, <h2>,.., <p> are block-level elements whereas <b> and <i> are inline elements.

5. CSS traditional box model

Reference: https://developer.mozilla.org/en-US/docs/Web/CSS/box_model

In an HTML document, each element is represented as a rectangular box. Determining the size, properties — like its color, background, borders aspect — and the position of these boxes is the goal of the browser.

This model describes the space taken by an element and around it. Each box has four areas: margin, border, padding, and content:



CSS provides properties to control each of these areas. We will explore some of them. The best way of learning these concepts is by experimenting.
(For details, see: https://developer.mozilla.org/en-US/docs/Web/CSS/box_model)

Margin area: Completely invisible - invisible on the browser screen. No background color. Transparent - will not obstruct objects behind it.
Border area: visible. You can change style and color.
Padding area: Area between content and border. This will have the same color as the element's background.
Content area: Visible with content.

We can control properties (color, style and width) of margin, border and padding areas. In fact, we can independently set different values for the top, right, bottom and left of these areas.

When you specify width and height of an element, you are actually specifying the width and height of the element itself (the innermost rectangle). The actual space taken by the box is the specified area plus the areas taken by padding, border and margin.

6. Specifying units of length in CSS property values

The following units of specification are used in CSS property values.

Unit keyword	Meaning	Usage example
pt	Point. One point is 1/72 inch	12pt
in	inch	
cm	centimeter	
mm	millimeter	
pc	Pica. One pica = 12 points	1pc
px	pixel	
em	current size	2em twice the current size
ex	1/2 of current size	

Note:

Sizes can be specified using %: 50% means 50% of the current size.

When you specify size using a unit, there should not be any space between the number and the unit: 12 pt is wrong, it should be 12pt.

Note:

1in = 96px, 3pt = 4px, and 25.4mm = 96px.

Note:

I commonly use px.

7. Specifying colors in CSS values

Colors are specified using two methods:

(1) Using three numbers R, G and B. These three numbers represent red (R), green (G) and blue (B). Each value is a number between 0 and 255. If you are not familiar with RGB representation of colors, see the Web (search for RGB colors). One nice site is, http://www.rapidtables.com/web/color/RGB_Color.htm.

(2) Using color keywords.

➢ Specifying a color using three numbers in CSS

- **Using decimal numbers: (r,g,b). Each of r, g, and b is a decimal number between 0 and 255.**

EXAMPLE:

(250,0,0) gives you red.

- **Using hexadecimal numbers: #RRGGBB. Each of R, G and B is a hexadecimal digit.**

EXAMPLE:

#FABB4F

- **Hexadecimal in the format #RGB. Each of R, G, and B is a hexadecimal digit.**

EXAMPLE:

#FB4

This is same as #FFBB44.

Note: Equal amounts of R, G, and B produces gray – of various grades: (0,0,0) to (255,255,255). (0, 0, 0) is white and (255,255,255) is black.

➢ **Representing colors using keywords**

You can specify a color using a keyword, such as red, blue or yellow.

The following is the original 16 keywords (along with hex and decimal values):

	black	#000000	0,0,0
	silver	#C0C0C0	192,192,192
	gray	#808080	128,128,128
	white	#FFFFFF	255,255,255
	maroon	#800000	128,0,0
	red	#FF0000	255,0,0
	purple	#800080	128,0,128
	fuchsia	#FF00FF	255,0,255
	green	#008000	0,128,0
	lime	#00FF00	0,255,0
	olive	#808000	128,128,0
	yellow	#FFFF00	255,255,0
	navy	#000080	0,0,128
	blue	#0000FF	0,0,255
	teal	#008080	0,128,128
	aqua	#00FFFF	0,255,255

Later "orange" (#FFA500) was added.

In CSS3 lot more keywords are included. See a long list here:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

Note:

In addition to using the usual RGB color values, CSS3 introduced what are called HSL (Hue, Saturation, and Lightness) values. We will not discuss HSL here.

8. CSS properties: width, height, min-width and min-height

You can set the width and height of an element by using the properties **width** and **height**.

- **Property width**

Syntax:

width : *value***;**

Where *value* is the width value in units described earlier.

- **Property height**

Syntax:

height : *value*;

Where *value* is the height value in units described earlier.

EXAMPLE:

```
width:100px;
height:20px;
```

- **Property min-height**

Syntax:

min-height : *value*;

Where *value* is the length value in units described earlier.

This property sets the minimum content height of a block.

An element to which min-height is applied will never be smaller than the minimum height specified, but will be allowed to grow normally if the content exceeds the minimum height set.

- **Property min-width**

Syntax:

min-width : *value*;

Where *value* is the length value in units described earlier.

This property sets the minimum content width of a block.

An element to which min-width is applied will never be narrower than the minimum width specified, but it will be allowed to grow normally if its content exceeds the minimum width set.

EXAMPLE:

min-width:100px;

min-height:20px;

Note:

The width and height are the dimensions of the innermost rectangle of the box (the element area). They do not include space taken by margin, border and padding.

9. CSS property: margin

The **margin** CSS property sets the margin for all four sides. There are five variations of **margin** property:

margin-top, **margin-right**, **margin-bottom** , **margin-left** and **margin**.

The values of these properties can be length in the usual units or percentages of the block.

The first four properties, **margin-top, margin-right, margin-bottom**, **and margin-left** control the margin on side of the box, as indicated by the name. The property **margin**, controls all the four sides.

EXAMPLES:

property : value	**Meaning**
margin-top : 15px;	Sets top margin to 15 pixels
margin-right : 15px;	Sets right margin to 15 pixels
margin-bottom : 15px;	Sets bottom margin to 15 pixels
margin-left : 15px;	Sets left margin to 15 pixels

The property **margin** can take 1 to 4 values:

- **margin with one value**

margin : *value;*

Sets all the four margins to the *value*.

EXAMPLE:

margin : 15px;

Sets all the four margins to 15px.

- **margin with two values**

margin : *value1 value2;*

Sets top and bottom margins to *value1* and right and left margins to *value2*.

EXAMPLE:

margin : 15px 10px;

Sets top and bottom margins to 15px and right and left margins to 10px.

- **margin with three values**

margin : *value1 value2 value3;*

Sets top margin to *value1*, right and left margins to *value2*, and bottom margin to *value3*.

EXAMPLE:

margin : 15px 10px 15px;

Sets top margin to 15px, right and left margins to 10px, and bottom margin to 15px.

- **margin with four values**

margin : *value1 value2 value3 value4;*

Sets top margin, right margin, bottom margin and left margin to *value1 value2*, *value3* and *value4* respectively.

EXAMPLE:

margin : 15px 10px 15px 10px;

Sets top margin, right margin, bottom margin and left margin to 15px, 10px, 15px and 10px respectively.

- **margin with value auto**

margin : auto;

The value auto sets the box is horizontally centered, 0 margin on top and bottom.

10. CSS property: padding

The **padding** CSS property sets the padding for all four sides. There are five variations of **padding** property:

padding-top, padding-right, padding-bottom , padding-left and padding.

The values of these properties can be length in the usual units or percentages of the block.

The first four properties, padding-top, padding-right, padding-bottom, and padding-left control the padding on side of the box, as indicated by the name. The property padding, controls all the four sides.

EXAMPLES:

property:value	**Meaning**
padding-top : 15px;	Sets top padding to 15 pixels
padding-right : 15px;	Sets right padding to 15 pixels
padding-bottom : 15px;	Sets bottom padding to 15 pixels
padding-left : 15px;	Sets left padding to 15 pixels

The property **padding** can take 1 to 4 values:

- **padding with one value**

padding : *value;*

Sets all the four paddings to the *value*.

EXAMPLE:

padding : 15px;

Sets all the four paddings to 15px.

- **padding with two values**

padding : *value1 value2;*

Sets top and bottom paddings to *value1* and right and left paddings to *value2*.

EXAMPLE:

padding : 15px 10px;

Sets top and bottom paddings to 15px and right and left paddings to 10px.

- **padding with three values**

padding : *value1 value2 value3;*

Sets top padding to *value1*, right and left paddings to *value2*, and bottom padding to *value3*.

EXAMPLE:

padding : 15px 10px 15px;

Sets top padding to 15px, right and left paddings to 10px, and bottom padding to 15px.

- **padding with four values**

padding : value1 value2 value3 value4;

Sets top padding, right padding, bottom padding and left padding to *value1 value2*, *value3* and *value4* respectively.

EXAMPLE:

padding : 15px 10px 15px 10px;

Sets top padding, right padding, bottom padding and left padding to 15px, 10px, 15px and 10px respectively.

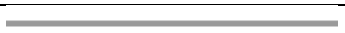
Note:

The property padding does not support **auto** value.

11. CSS property: border-style

The following CSS properties set the styles of the lines of the border of an element. The following are the possible values for border styles:

border style value	Meaning	Output in Chrome
none	no border	
hidden	no border	
dotted	series of rounded dots	
dashed	series of dashes	
solid	solid line	
double	two solid lines	
groove	a line with carved effect	
ridge	a line with 3d effect	

inset	a line which looks embedded	
outset	a line which looks bulged	

The **border-style** CSS property sets the border-style for all four sides. There are five variations of **border-style** property:

border-top-style, border-right-style, border-bottom-style, border-left-style and border-style.

The values of these properties can be border-style values from the table above.

The first four properties **border-top-style**, **border-right-style**, **border-bottom-style**, and **border-left-style** control the border-style on side of the box, as indicated by the name. The property **border-style**, controls all the four sides.

EXAMPLES:

property : value	**Meaning**
border-top-style: **solid**;	Sets top border to solid line
border-right-style: **dotted**;	Sets right border to dotted line
border-bottom-style: **double**;	Sets bottom border to double line
border-left-style: **dashed**;	Sets left border to dashed line

The property **border-style** can take 1 to 4 values:

- **border-style with one value**

border-style : *value;*
Sets all the four border-styles to the *value*.

EXAMPLE:

border-style : solid;
Sets all the four border-styles to solid.

- **border-style with two values**

border-style : *value1 value2;*
Sets top and bottom border-styles to *value1* and right and left border-styles to *value2*.

EXAMPLE:

border-style : solid dotted;

Sets top and bottom border-styles to solid and right and left border-styles to dotted.

- **border-style with three values**

border-style : *value1 value2 value3;*

Sets top border-style to *value1*, right and left border-styles to *value2*, and bottom border-style to *value3*.

EXAMPLE:

border-style : dotted solid double;

Sets top border-style to dotted, right and left border-styles to solid, and bottom border-style to double.

- **border-style with four values**

border-style : *value1 value2 value3 value4;*

Sets top border-style, right border-style, bottom border-style and left border-style to *value1 value2*, *value3* and *value4* respectively.

EXAMPLE:

border-style : solid double dotted double;

Sets top border-style, right border-style, bottom border-style and left border-style to solid, double, dotted, and double respectively.

12. CSS property: border-color

The **border-color** CSS property sets the border-color for all four sides. Border-style has to be set for border-color to work.

There are five variations of **border-color** property:

border-top-color, border-right-color, border-bottom-color, border-left-color and border-color.

The first four properties **border-top-color**, **border-right-color**, **border-bottom-color**, and **border-left-color** control the border-color on side of the box, as indicated by the name. The property **border-color**, controls all the four sides.

The values of these properties are color values:

Property : value	**Meaning**
border-top-color: *colorValue*	Sets the top border color
border-right-color: *colorValue*	Sets the right border color
border-bottom-color: *colorValue*	Sets the bottom border color
border-left-color: *colorValue*	Sets the left border color

The property **border-color** can take 1 to 4 values:

- **border-color with one value**

border-color : *colorValue;*
Sets all the four border-colors to the *value*.

EXAMPLE:

border-color : blue;
Sets all the four border-colors to blue.

- **border-color with two values**

border-color : *value1 value2;*
Sets top and bottom border-colors to *value1* and right and left border-colors to *value2*.

EXAMPLE:

border-color : blue red;
Sets top and bottom border-colors to blue and right and left border-colors to red.

- **border-color with three values**

border-color : *value1 value2 value3;*
Sets top border-color to *value1*, right and left border-colors to *value2*, and bottom border-color to *value3*.

EXAMPLE:

border-color : red blue yellow;

Sets top border-color to red, right and left border-colors to blue, and bottom border-color to yellow.

- **border-color with four values**

border-color : *value1 value2 value 3 value4;*

Sets top border-color, right border-color, bottom border-color and left border-color to *value1, value2, value 3, and value4* respectively.

EXAMPLE:

border-color : *red blue orange yellow;*

Sets top border-color, right border-color, bottom border-color and left border-color to red, blue, orange, and yellow respectively.

13. CSS property: border-width

The **border-width** CSS property sets the border-width for all four sides. There are five variations of **border-width** property:

border-top-width, **border-right-width**, **border-bottom-width**, **border-left-width** and **border-width**.

The first four properties **border-top-width**, **border-right-width**, **border-bottom-width**, and **border-left-width** control the border-widths on sides of the box, as indicated by the name. The property **border-width**, controls all the four sides.

The values of these properties are length in usual units. Border width can also be specified using the keywords: **thin**, **medium** and **thick**. The specification doesn't precisely define the thickness of each of the keywords, which is therefore implementation specific. Nevertheless, it requests that the thickness does follow the thin ≤ medium ≤ thick inequality and that the values are constant on a single document.

Property	Meaning
border-top-width: *value*	Sets the top border thickness to *value*

border-right-width: *value*	Sets the right border thickness to *value*
border-bottom-width: *value*	Sets the bottom border thickness to *value*
border-left-width: *value*	Sets the left border thickness to *value*

The property **border-width** can take 1 to 4 values:

- **border-width with one value**

border-width : *value;*
Sets all the four border-widths to the *value*.

EXAMPLE:

border-width : 18px;
Sets all the four border-widths to 18 pixels.

- **border-width with two values**

border-width : *value1 value2;*
Sets top and bottom border widths to *value1* and right and left border widths to *value2*.

EXAMPLE:

border-width : 15px thick;
Sets top and bottom border widths to 15 pixels and right and left border widths to thick.

- **border-width with three values**

border-width : *value1 value2 value3;*
Sets top border-width to *value1*, right and left border-widths to *value2*, and bottom border-width to *value3*.

EXAMPLE:

border-width : 10px 15px 10px;
Sets top border width to 10 pixels, right and left border widths to 15 pixels, and bottom border width to 10 pixels.

- **border-width with four values**

border-width : *value1 value2 value 3 value4;*
Sets top border-width, right border-width, bottom border-width and left border-width to value1, value2, value 3, and value4 respectively.

EXAMPLE:

border-width : *10px 15px 10px 15px;*
Sets top border-width, right border-width, bottom border-width and left border-width to *10px, 15px, 10px, and 15px* respectively.

---

14. CSS Property: outline

The CSS property outline helps us create outlines around visual objects such as buttons, active form fields, image maps, etc., to make them stand out.

Outlines differ from borders in the following ways:

- Outlines do not take up space.
- Outlines may be non-rectangular.

The **outline** property can be used to control the styles, colors, and widths of outlines.

➢ The outline-style property

The **outline-style** property accepts the same values (none, dotted, dashed, solid, double, etc.) as **border-style**. (Except, 'hidden' is not a legal outline style.)

➢ The outline-color property

The **outline-color** accepts all color values, as well as the keyword 'invert'. 'Invert' performs a color inversion on the pixels on the screen. This is a common trick to ensure the focus border is visible, regardless of color background.

➢ The outline-width property

The **outline-width** property accepts the same values as **border-width**.

➢ The outline property

The **outline** property is a shorthand property, and sets all three of **outline-style, outline-width,** and **outline-color**.

NOTE:

The outline is the same on all sides. In contrast to borders, there is no **outline-top**, **outline-left, etc.,** properties.

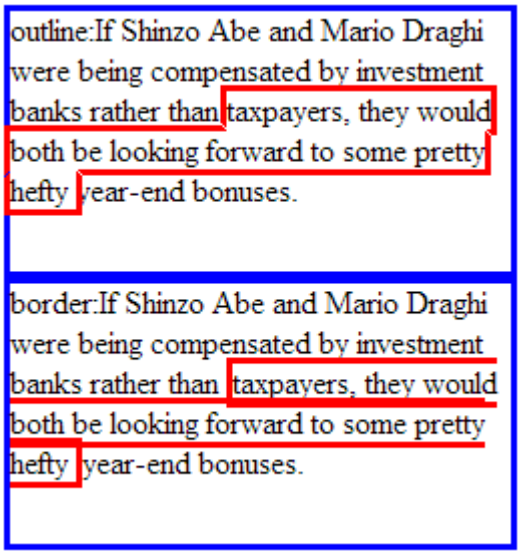
Example:

To understand the difference between how the properties outlie and border work, see this example.

```
div{
    width:250px;
    height:130px;
    border:solid blue;
}
#one{
outline:solid red;
}
#two{
border: solid red;
}
</style>
<body>
  <div><This is <b>outline</b>:If Shinzo Abe and Mario Draghi were being
compensated by investment banks rather than <span id="one">taxpayers,
  they would both be looking forward to some pretty hefty </span>
  year-end bonuses.</div>

<div><This is <b>border</b>:If Shinzo Abe and Mario Draghi were being
compensated by investment banks rather than <span id="two">taxpayers,
  they would both be looking forward to some pretty hefty </span>
  year-end bonuses.</div>
</body>
```

Result:



15. Setting text features

These properties effect the text content of the element.

- **The color property**

The CSS **color** property sets the text color of the element's content.

For the color property value, you can either use the color keywords or the RGB values (as described earlier).

Examples:

color : blue;

color : #ffaa00;

color : rgb(200,150,200);

- **The letter-spacing property**

The **letter-spacing** property sets extra spacing between characters in the text content of an element.

Syntax:

letter-spacing : *value*;

Where value is one of the following values:

Value	Meaning
normal	Normal spacing
length	A number in the usual length units. Examples: 1px, 1em, 0,5em.

- **The word-spacing property**

The **word-spacing** property sets extra spacing between words in the text content of an element.

Syntax:
word-spacing : *value*;

Where *value* is one of the following values:

Value	Meaning
normal	Normal spacing
length	A number in the usual length units. Examples: 1px, 1em, 0,5em.

- **The line-height property**

The **line-height** property sets extra spacing between lines in the text content of an element. (That is, effects line spacing.)

Syntax:
line-height : *value*;

Where *value* is one of the following values:

Value	Meaning
normal	Normal spacing
length	A number in the usual length units. Examples: 1px, 1em, 0,5em.
number	Just a number. Produces multiple of the number.

	Example: 2 means twice the current spacing.

- **The text-align property**

The **text-align** property specifies how the inline content of a block is aligned.

Syntax:
text-align : *value*;

Where *value* is one of the following values:

Value	Meaning
center	Text is centered
justify	Text is left and right justified
left	Text is left justified
right	Text is right justified

- **The text-decoration property**

The **text-decoration** property text formatting to underline, overline, line-through or blink.

Syntax:
text-decoration : *value*;

Where *value* is one of the following values:

Value	Meaning
blink	Blinking text
line-through	Draws a horizontal line through the text.
overline	Draws a horizontal line above the text.
underline	Draws a horizontal line below the text.

Note: Browsers do not seem to support blink value.

- **The text-indent property**

The **text-indent** property specifies the indentation of the first line of text in a block.

Syntax:
text-indent : *value*;

Where *value* is a number in usual units.

- **The text-transform property**

The **text-transform** property specifies if the text needs to be capitalized.

Syntax:
text-transform : *value*;

Where *value* is one of the following:

Value	Meaning
capitalize	Transforms the first character in each word to uppercase; all other characters remain unaffected—they're not transformed to lowercase, but will appear as written in the document.
lowercase	Transforms all characters to lowercase.
uppercase	Transforms all characters to uppercase.
none	No effect

Example:

```
<style>
  .one{text-transform: capitalize}
  .two{text-transform: lowercase}
  .three{text-transform: uppercase}
 </style>


 <h3 class="one">new york city</h1>
 <h3 class="two">New York City</h1>
```

```
<h3 class="three">New York City</h1>
```

Chrome output:

New York City

new york city

NEW YORK CITY

- **The text-shadow property**

The **text-shadow** property adds shadows to text. It accepts a comma-separated list of shadows to be applied to the text.

Each shadow is specified as an offset from the text, along with optional color and blur radius values.

Multiple shadows are applied front-to-back, with the first-specified shadow on top.

Syntax:
text-shadow : *value*;

Where *value* is one of the following:

Value	Meaning
color	Optional. Color value in the usual format. This specifies the shadow color.
offset-x offset-y	Required. Two lengths (in the usual units) specifying horizontal and vertical distances respectively.
blur-radius	Optional. This is a length in the usual units. The higher this value, the bigger the blur; the shadow becomes wider and lighter.

Example:

```
<style>
  .one {color:white; text-shadow: red 2px 2px 3px;}
  .two {color:black;text-shadow: blue 3px 2px 2px;}
  .three {color:red;text-shadow: green 2px 2px 3px;}
 </style>

<body >
 <h1 class="one">New York City</h1>
 <h1 class="two">New York City</h1>
 <h1 class="three">New York City</h1>
 </body>
```

Chrome output:

16. Setting font features

CSS provides several properties to set font features of the text in an element:

- **The font-family property**

Syntax:

font-family : *value;*

The value is really a prioritized list of font family names and/or generic family names to be specified for the selected element. Font family values are separated by a comma to indicate that they are alternatives.

If the first font on the list is not available on the computer, the next font on the list will be tried until a suitable font is found.

There are two types of names used to categorize fonts: generic families and non-generic families. The two terms are explained below.

Generic family names

There are five types of generic font families. Here is a list of those:

- serif
- sans-serif
- cursive
- fantasy
- monospace

Each represents a group of fonts.

Examples of Serif fonts (fonts which "contain feet."): Times, Times New Roman, Palatino, Bookman, New Century, Georgia, and Schoolbook.

Examples of Sans-Serif fonts (fonts which "do not contain feet."): Arial, Helvetica, Gill Sans, Lucida, Verdana, and Helvetica Narrow.

Examples of Monospace fonts: Andale Mono, Courier New, Courier, Lucidatypewriter, and Fixed.

Examples of Cursive fonts: Comic Sans, Comic Sans MS, Zapf Chancery, Coronetscript, Florence, Brush Script MT, and Parkavenue.

Examples of Fantasy fonts: Impact, Arnoldboecklin, Oldtown, Blippo, and Brushstroke.

Non-generic family names:

There are many. Examples of non-generic family names (often known as "font") are "Arial", "Times New Roman" or "Tahoma".

- **The font-size property**

Syntax:

font-size: *value;*

The value can be one of the following:

Value	Meaning
xx-small **x-small** **small** **medium** **large** **x-large** **xx-large**	These are called absolute size. The exact sizes to which those keywords map aren't defined. You need to experiment with them.
smaller **larger**	These are called relative sizes. They make the font size smaller or larger than the inherited value. You need to experiment with them.
length	A number using the usual units. You can also use %.

- **The font-style property**

Syntax:

font-style: *value;*

Where value can be one of the following:

Value	Meaning
italic	italic font
normal	Normal font (upright)
oblique	Slanted font (similar to italic)

- **The font-weight property**

The font-weight property specifies the weight or boldness of the font.

Syntax:

font-weight: *value;*

Where value can be one of the following:

Value	Meaning
normal	Normal font

bold	bolder than normal
lighter	Lighter than normal
bolder	Darker than the parent element
A number	100, 200, 300, 400, 500, 600, 700, 800, 900 Numeric font weights for fonts that provide more than just normal and bold. Experiment with using these numbers.

- **The font-variant property**

The **font-variant** property specifies whether to use normal font or small-caps.

Syntax:

font-variant: *value;*

Where value can be one of the following:

Value	**Meaning**
normal	Normal font
small-caps	Specifies a font that is labeled as a small-caps font. If a small-caps font is not available, Mozilla (Firefox) and other browsers will simulate a small-caps font, i.e. by taking a normal font and replacing the lowercase letters by scaled uppercase characters.

- **The font property**

The **font** property sets the font size and the font family, plus, optionally, the font color, font variant, font weight, and line height, for an element's text content.

Syntax:

font: *value;*

Where value is a list of font size and the font family, plus, optionally, the font color, font variant, font weight, and line height, for an element's text content.

Using additional cool fonts

You will have a limited number of fonts installed on your computer. To make your Web pages dazzling you may need cool dazzling fonts. There are two ways you can use fonts which are currently not on your computer:

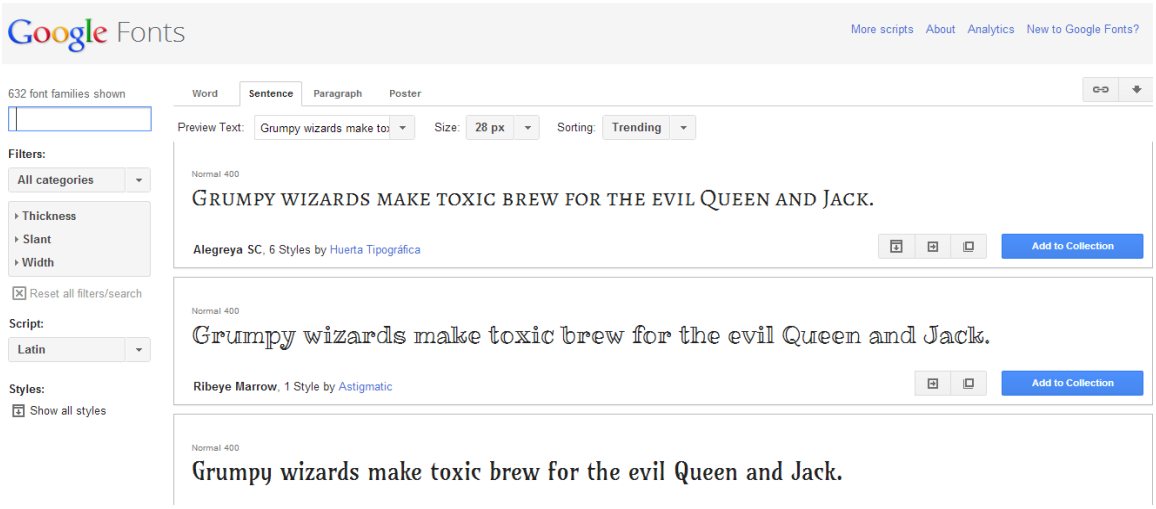
1. You can use fonts which are available online. You can use them directly from your Web page without installing the fonts on your machine. We will see examples below.
2. You can find fonts available (free or buy them) from the Web. You can download them and make them available to your Web pages. We will see examples below.

Using fonts available online without downloading them

There are plenty Web sites, which provide fonts online for your use. Let me explain how to use one of them in your Web page with an example:

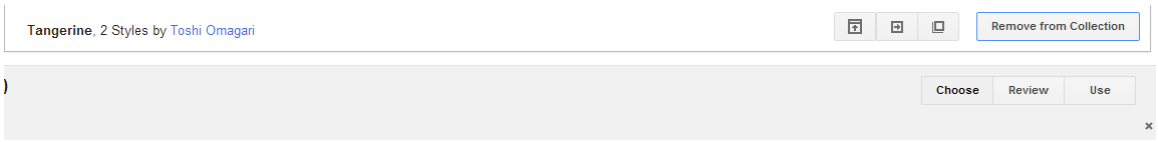
Go to the Web site which provides online fonts (free). Here is one:
http://www.google.com/fonts

This site gives a long list of fonts available to you for free:



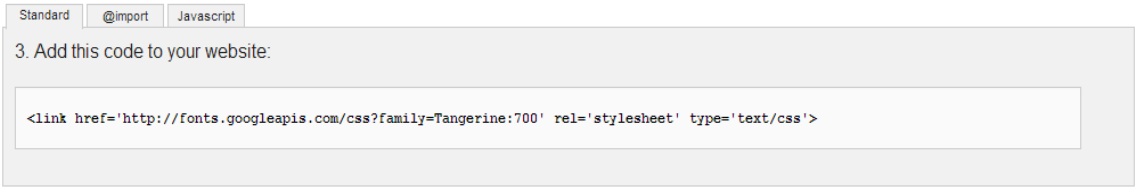
Scroll down and see the font you like. I will choose, Tangerine (that is the example Google gives):

Click Add to Collection button. You will see this screen:



Click **Use** button.

On the next screen, scroll down to see the URL for the font:



We need this URL.

Also, on the same page, notice the syntax for using **font-family** CSS property:

We need this as well.

Now incorporating this font in our HTML page:

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Tangerine">
  <style>
   body {
    font-family: 'Tangerine', serif;
    font-size: 48px;
   }
  </style>
 </head>
<body>
 New York New York New York
</body>
</html>
```

Chrome output:

New York New York New York

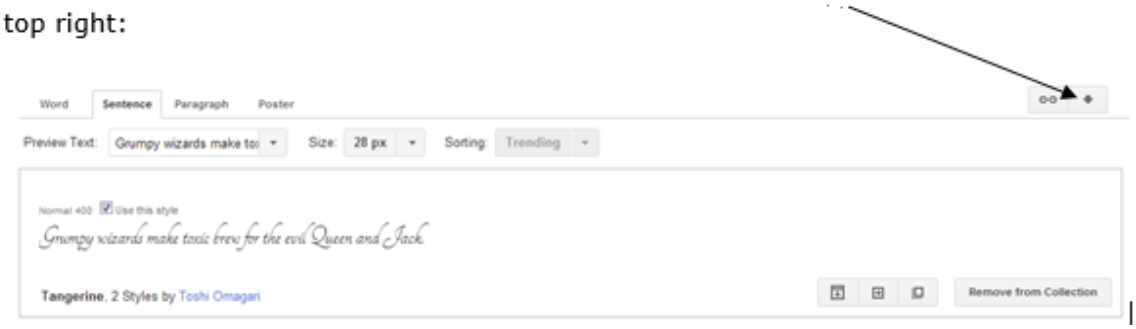
Downloading fonts and using them

Instead of accessing through their URL, you can download and save the font on your computer and use it. Here is how you do it with an example:

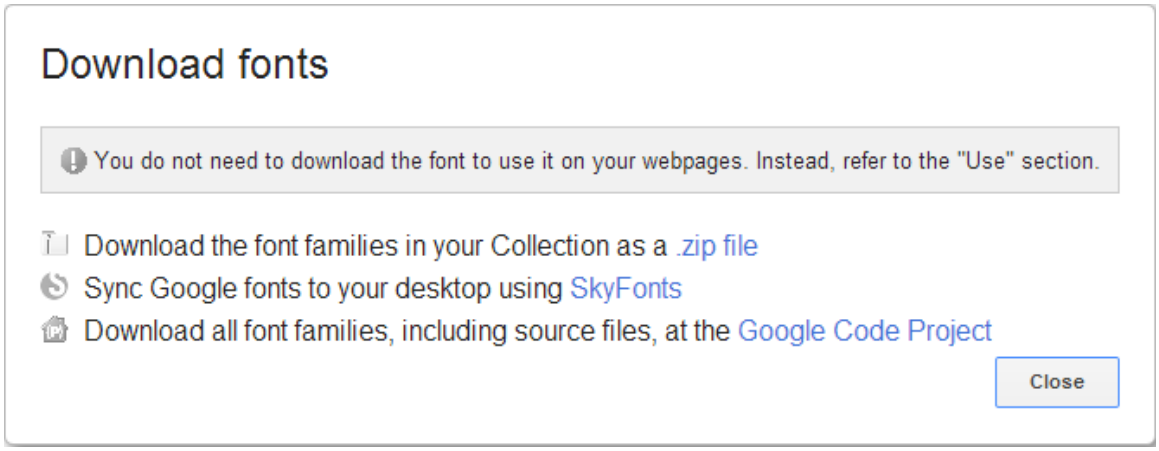
As before, select the font you like on the Google site: http://www.google.com/fonts.

Let us select the same font, Tangerine:

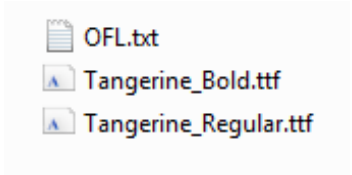
Click Add to Collection button. Now notice that there is a down arrow button at the top right:



On the pop-screen,



Click **.zip file**. This will download the font file, Tangerine.zip. Unzip it to see three files,



You need the two ttf files. Copy the two files to your directory where you keep your HTML files.

You are now ready to use the font. To use your own fonts in CSS, you need to use the keyword **@font-face**. Here is a sample HTML code:

```
<html>
<head>
  <style>
   @font-face
   {
   font-family: nmfont;
   src: url(Tangerine_Bold.ttf);
   font-size: 48px;
   }
  body {
     font-family: nmfont;
```

```
    font-size: 48px;
  }
  </style>
 </head>
<body>
 New York New York New York
</body>
</html>
```

NOTE:

Rememember the syntax for specifying the new font:

```
@font-face
   {
   font-family: nmfont;
   src: url(Tangerine_Bold.ttf);
   font-size: 48px;
   }
  body {
    font-family: nmfont;
    font-size: 48px;
  }
```

Here **nmfont** is just a name made by me. You can use any name. The same name must be used in both @font-face{…} and CSS definition (body tag in our example).

Note: You can define more than one @font-face with different font-family names for different tags.

Other Web sites I have found for fonts:

http://www.1001freefonts.com/

http://www.fontsquirrel.com/fonts/list/hot

http://www.fontex.org/
http://www.fonts.com/web-fonts
http://scripts.sil.org/cms/scripts/page.php?cat_id=FontDownloads
http://www.smashingmagazine.com/2007/11/08/40-excellent-freefonts-for-professional-design/
http://openfontlibrary.org/en/catalogue
http://www.adobe.com/products/type/font-licensing/licensing-faq.html (fee)
http://www.upsdell.com/BrowserNews/res_fonts.htm#a04
http://hellohappy.org/beautiful-web-type/
http://fonts.simplythebest.net/

NOTE:

A comment on font file extensions and browser support:

- Internet Explorer only supports EOT
- Mozilla browsers support OTF and TTF
- Safari and Opera support OTF, TTF and SVG
- Chrome supports TTF and SVG.
- Mobile browsers like Safari on the iPad and iPhone require SVG.

EOT = Embedded Open Type
OTF = Open Type Font
TTF = True Type Font
SVG = Scalable Vector Graphics

17. Setting background features

CSS provides several properties to set background features of an element: We will introduce five background related properties: **background-color**, **background-image, background-repeat, background-attachment, background-size, background-position, and** the combine all property **background.**

Remember "background" refers to the background of the content, padding and border areas. Border colors and styles are set with the border properties. Margins are always transparent.

- **Property background-color**

Syntax:

background-color : *value*;

Where *value* is the color value (as described earlier).
This sets the background color of the element.

- **Property background-image**

Syntax:

background-image : *value*;

Where *value* is the image filename in the format **url**("*filename*").
This sets the background image for the element. Even if the image size is small, it will be repeated throughout to cover the whole element area.

- **Property background-repeat**

This property is used along with **background-image** property.

Syntax:

background-repeat : *value*;

Where *value* is one of the repeat values:

Value	Meaning
repeat-x	The image is repeated in x direction as much as needed to cover the whole background image painting area. The last image may be clipped if the whole thing won't fit in the remaining area.
repeat-x	The image is repeated in y direction as much as needed to cover the whole background image painting area. The last image may be clipped if the whole thing won't fit in the remaining area.
repeat	The image is repeated in both x and y directions as much as needed to cover the whole background image painting area. The last image may be clipped if the whole thing won't fit in the remaining area. This is the default setting.
no-repeat	The image is not repeated (and hence the background image painting area will not necessarily been entirely covered).

- **Property background-attachment**

This property is used along with **background-image** property. The **background-attachment** property determines whether that image's position is fixed within the viewport, or scrolls along with its containing block.

Syntax:

background-attachment : *value*;

Where *value* is one of the values:

Value	**Meaning**
scroll	The background image will scroll within the viewport along with the block the image is contained within.
fixed	The background image will not scroll with its containing element, instead remaining stationary within the viewport.
local	The background image will not scroll with its containing element, but will scroll when the element's content scrolls: it is fixed regarding the element's content.

- **Property background-size**

This property is used along with **background-image** property. The **background-size** property determines the size of the background image.

Syntax:

background-size : *value*;

Where value is one of the values:

Value	Meaning
auto	Scales the background image in the corresponding direction such that its intrinsic proportion is maintained.
cover	Scales the background image to be as small as possible while ensuring both its dimensions are greater than or equal to the corresponding dimensions of the background positioning area.

contain	Specifies that the background image should be scaled to be as large as possible while ensuring both its dimensions are less than or equal to the corresponding dimensions of the background positioning area.
length	A number in length units described earlier.
percentage	Scales the background image in the corresponding dimension to the specified percentage of the background positioning area.

- **Property background-position**

This property is used along with **background-image** property. The **background-position** property determines the position of the background image.

Syntax:

background-position : *value*;

Where value can be specified in several ways, as shown by the following examples: left, right, center, bottom, 25% 75%, 2cm bottom, right 20px bottom 20px.

- **Property background**

The **background** property is a shorthand property for setting the individual background properties. You can use background to set background-color, background-image, background-repeat, background-attachment and background-position.

Example:

background : blue;

This sets background to blue.

18. Managing lists

The **list-style** property provides features to control list item appearance, whether to include an image with the list item, and positioning of the list item.

There are four variations:

list-style-type

list-style-image

list-style-postion

list-style

- **The list-style-type property**

This property controls the appearance of the list items.

Some of the supported values:

In the following the output column displays output with different values for the following code:

```
<style>
li { list-style-type: value ;}
</style>

<nav>
   <ul>
   <li><a href="http://microsoft.com">Microsoft</a></li>
   <li><a href="http://google.com">Google</a></li>
   <li><a href="http://ibm.com">IBM</a></li>
   <li><a href="http://cisco.com">Cisco</a></li>
   <li><a href="https://developer.mozilla.org">Mozilla</a></li>
   </ul>
</nav>
```

Value	Meaning	output
list-style-type: disc;	A filled circle (default value)	• Microsoft • Google • IBM • Cisco • Mozilla
list-style-type: circle	A hollow circle	○ Microsoft ○ Google ○ IBM ○ Cisco ○ Mozilla

list-style-type: square	A filled square	▪ Microsoft ▪ Google ▪ IBM ▪ Cisco ▪ Mozilla
list-style-type: decimal	numbers starting from 1	1. Microsoft 2. Google 3. IBM 4. Cisco 5. Mozilla
list-style-type: lower-roman	Lowercase roman numerals: i, ii, iii	i. Microsoft ii. Google iii. IBM iv. Cisco v. Mozilla
list-style-type: upper-roman	Uppercase roman numerals: I, II, III	I. Microsoft II. Google III. IBM IV. Cisco V. Mozilla
list-style-type: lower-alpha	Lowercase letters: a, b, c	a. Microsoft b. Google c. IBM d. Cisco e. Mozilla
list-style-type: upper-alpha	Uppercase letters: A, B, C	A. Microsoft B. Google C. IBM D. Cisco E. Mozilla

lower-greek	Lower Greek letters	α. Microsoft β. Google γ. IBM δ. Cisco ε. Mozilla
Hebrew	Hebrew letters	א. Microsoft ב. Google ג. IBM ד. Cisco ה. Mozilla
none	none	Microsoft Google IBM Cisco Mozilla

Note: There are many other types. See https://developer.mozilla.org/en-US/docs/Web/CSS/list-style-type for more info.

- **The list-style-image property**

The list-style-image CSS property sets the image that will be used as the list item marker.

The value is an image in the syntax:
list-style-image : url("imagefile");

- **The list-style-position property**

The list-style-position CSS property specifies the position of the marker box in the principal block box.

Values:

Value	Meaning
outside	The marker box is outside the principal block box.
inside	The marker box is the first inline box in the principal block box, after which the element's content flows.

- **The list-style property**

The list-style CSS property is a shorthand property for setting list-style-type,list-style-image and list-style-position.

EXAMPLES:

list-style: circle;

list-style: circle inside;

NOTE:

Normally the and <ol> are included within <nav> … </nav> tag. This will provide more control over the lists.

EXAMPLE:

Generally list items are displayed one below other. The display property with value inline displays the list horizontal:

```
li{list-style-type: none ;display:inline;}
```

OUTPUT:

Microsoft Google IBM Cisco Mozilla

Example:

Making the list colorful.

```
li{
list-style-type: none ;
display:inline;
background-color: cyan;
width: 60px;
border: 1px solid #fff;
border-syle: 10px 10px 10px 10px;
}
a{color:blue;}
```

Output:

Note: If you remove the **display** property, list appears one below the other.

19. Pseudo-Elements

Pseudo-Elements are used to make property affect only part of the content of the specified selector.

Syntax:

selector::*pseudo-element*{*property*:*value*; …}

The *pseudo-element* specifies that the specified property is to be applied to the content under the *selector*.

Important pseudo-elements

- ::first-letter
- ::first-line
- ::after
- ::before

Note: You can use either :: or :.

➢ **The pseudo-element ::first-letter**

Syntax:

selector::**first-letter**{*property : value;….}*

The **first-letter** pseudo element is used only with block-level selectors. The specified property will be applied to the first character in the block.

NOTE:

::first-letter does not work on inline elements such as a span. ::first-letter works on block elements such as a paragraph, table caption, table cell, list item, or those with the inline-block property applied.

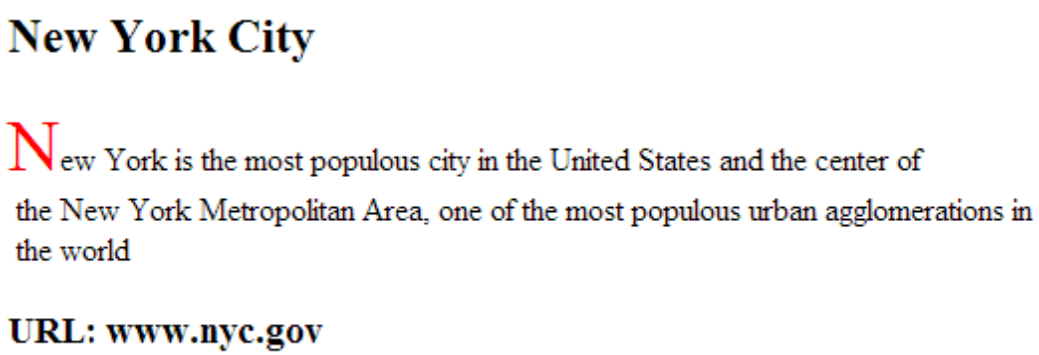
Therefore it's better to apply ::first-letter to a p instead of a span.

p::first-letter {font-size: 500px;}

Example:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
  div {white-space: pre;}
  div::first-letter{font-size:36px;color:red;}
 </syle>
 </head>
 <body >
 <h2>New York City</h2>
 <div>New York is the most populous city in the United States and the center of
 the New York Metropolitan Area, one of the most populous urban agglomerations in
 the world</div>
 <h3>URL: www.nyc.gov</h3>
 </body>
</html>
```

Chrome output:



- **The pseudo-element::first-line**

Syntax:

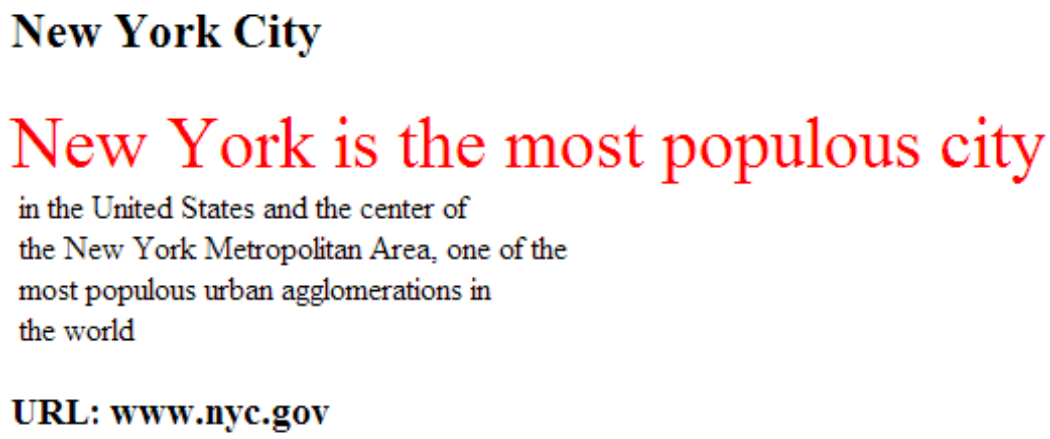
selector::**first-line**{*property:value;….}*

The **first-line** pseudo element is used only with block-level selectors. The specified property will be applied to the first line in the block.

Example:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
   div {white-space: pre;}
   div::first-line{font-size:36px;color:red;}
 </style>
 </head>
 <body >
 <h2>New York City</h2>
 <div>New York is the most populous city
 in the United States and the center of
 the New York Metropolitan Area, one of the
 most populous urban agglomerations in
 the world</div>
 <h3>URL: www.nyc.gov</h3>
 </body>
</html>
```

Chrome output:



- **The pseudo-element ::after**

Syntax:

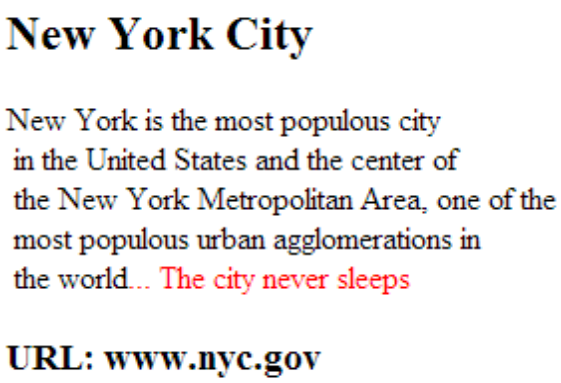
*selector***:after**{ }

You need to use the **content** property with the pseudo element **after**, as shown in the example below:

EXAMPLE:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
  div {white-space: pre;}
  div::after{content:"... The city never sleeps";color:red;}
 </style>
 </head>
 <body >
 <h2>New York City</h2>
 <div>New York is the most populous city
 in the United States and the center of
 the New York Metropolitan Area, one of the
 most populous urban agglomerations in
 the world</div>
 <h3>URL: www.nyc.gov</h3>
 </body>
</html>
```

Chrome output:



- **The pseudo-element ::before**

Syntax:

selector:**before**{ }

You need to use the **content** property with the pseudo element **before**, as shown in the example below:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
  div {white-space: pre;}
  div::before{content:"The city thet never sleeps ...";color:red;}
 </style>
 </head>
 <body >
 <h2>New York City</h2>
 <div>New York is the most populous city
 in the United States and the center of
 the New York Metropolitan Area, one of the
 most populous urban agglomerations in
 the world</div>
 <h3>URL: www.nyc.gov</h3>
 </body>
</html>
```

New York City

The city thet never sleeps ...New York is the most populous city
in the United States and the center of
the New York Metropolitan Area, one of the
most populous urban agglomerations in
the world

URL: www.nyc.gov

---

20. The float and clear properties

A browser renders a Web page by keeping the block elements one below the other, aligning them (by default) on left. This is the norm flow of block elements. After placing an element on the screen, even if there is space on the right side of the block, the next element is placed beneath this element.

The CSS property **float** is used to make elements appear one next to the other instead of one below the other. When an element is associated with a CSS declaration **float:left;**, the element moves to the left in its usual position and makes room for the next element to come next to it on the right.

EXAMPLE 1:

```
<style>
#one { width:40px; height:40px; background-color:green;}
</style>

<body>
<div id="one">One</div>
<p>New York City</P>
</body>
```

Normal flow is:

One

New York City

EXAMPLE 2:

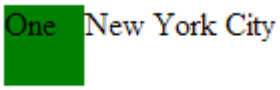
Let us add float:left to #one.

```
<style>
#one {float:left;width:40px; height:40px; background-color:green;}
</style>

<body>
<div id="one">One</div>
<p>New York City</P>
</body>
```

Because of float:left, the element following One comes next to this.

Output:



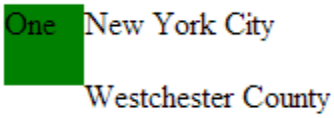
EXAMPLE 3:

Let us add one mode element to the HTML document,

```
<style>
#one {float:left;width:40px; height:40px; background-color:green;}
</style>

<body>
<div id="one">One</div>
<p>New York City</P>
<p>Westchester County</p>
</body>
```

Because there is room on the right side, the next element is also placed there:



EXAMPLE 4:

If you want an element to appear on the right, but to follow the normal flow of appearing below, you have to use **clear** property, with value **both**.

```
 <style>
#one {float:left;width:40px; height:40px; background-color:green;}
#two {clear:both;}
</style>

<body>
<div id="one">One</div>
<p>New York City</P>
<p id="two">Westchester County</p>
</body>
```

NOTE:

Same ideas hold good for **float=right.**

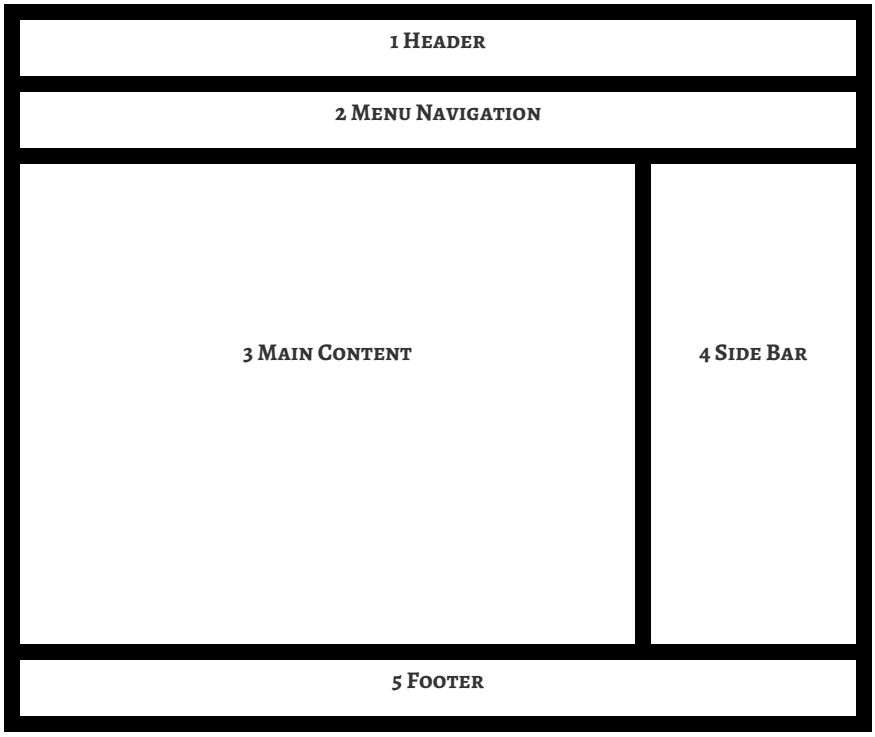
21. The display property

The display CSS property specifies the type of rendering box used for an element. Important values: block or in-line.

22. CSS layout – a general example – Two Column Web Page

This example explores a simple Web site layout design using CSS. Obviously, creating an attractive Web site is an art. This example gives you general steps to design a Web site.

First you must identify the main structural elements of the design. We will adopt the box model. That is, the content of every element is enclosed in a rectangle. When in doubt use the tag.
Our example will have the following structure:



The following are not rules – but one set of suggestions. You have to modify them to fit your needs and style.

In my CSS style sheet, I first declare several @font-face for various fonts. I call them, nmfont, nmfont1, nmfont2, and nmfont3.

1. **START YOUR <BODY>:**

```
<body>
  <div id="container">
    …
    …
    …
  </div>
</body>
```

WE WILL HAVE A <DIV> TAG FOR EACH OF THE 5 CONTENTS ELEMENTS:

```
<body>
  <div id="container">
     <div id="header"></div>
     <div id="menu"></div>
     <div id="main"></div>
     <div id="sidebar"></div>
     <div id="footer"></div>
  </div>
</body>
```

TYPICAL #CONTAINER CSS RULE:

```
#container {
  margin:auto;
  width: 760px;
  background: #50d07d;
}
```

This is the width of the area seen on the browser. Developers make this width 760px – 960px. The background color covers the entire area. This is the color you see when no other rectangle covers the container.

INCLUDE CSS RULES FOR <BODY> AND <HTML>

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

HEADER CSS

```
#header{
   background-color : #d02552;
   height : 150px;
```

```
  text-align : center;
  padding-top : 25px;
  font-family : nmfont;
  font-size : 35px;
}
```

The property padding-top is included to bring the text vertically down.
This is how it looks in Chrome:

MENU CSS

```
#menu {
    background-color : #b2cecf;
    height : 40px;
    font-family : nmfont3;
    font-size : 20px;
    padding:15px;
    text-align : center;
}
#menu li{
    display:inline;
}
```

In the menu navigation content I have included three links. To make the links appear horizontally, I have included display : inline in CSS.
This is how it looks in Chrome:

MAIN CONTENT CSS

```
#main{
```

```
    float:left;
    width : 335px;
    height:360px;
    background-color: orange;
    padding : 20px;
    font-family : nmfont1;
    font-size : 18px;  }
```

This is the left column. I have made the width slight less than total width (760px). The right column (sidebar) will be the same. I have made the height 360px. The sidebar will be the same height. This column is float left and the sidebar will be float right.

Let us see the look in Chrome after we write the right column.

SIDEBAR CSS

```
#sidebar{
    float : right;
    width : 335px;
    height: 360px;
    background: #d025FF;
    padding:20px;
    font-family : nmfont1;
    font-size : 18px;
}
```

Notice that characteristics of left and right columns are the same, except float.

This how they together look in Chrome:

Introduction to three topics important in Web development for modern browsers: HTML5, CSS3 and JQuery. The Web technology is changing rapidly. HTML5, CSS3 and JavaScript are providing Web developers new tools to create amazing Web sites. No assumption is made on the background of students. All topics are started from the beginning. Practical examples are the focus. This is a project-based course, in which students will do hands-on projects every week. The combination of HTML5, CSS3 and JavaScript allows students to create attractive, feature-rich, and dynamic Web sites using these latest Web technologies.

Your responsibility every week includes:
Reading materials assigned from the textbook
Reading my notes
Taking a quiz: Quizzes are online and are timed (generally 30 minutes).

Evaluation

Each weekly online quiz is worth 10 points

Each Weekly Lab is worth 15 points

No other exams.

Late penalties:

Assignment submitted late 5 points/week

Quizzes taken late 5 points/week

Will not accept submissions more than one week late.

FOOTER CSS

```
#footer{
    clear : both;
    padding-top : 20px;
    color : white;
    background-color: black;
    height : 40px;
    text-align : center;
}
```

Notice the **clear** property. Inclusion of this is important.

This is how it looks in Chrome:

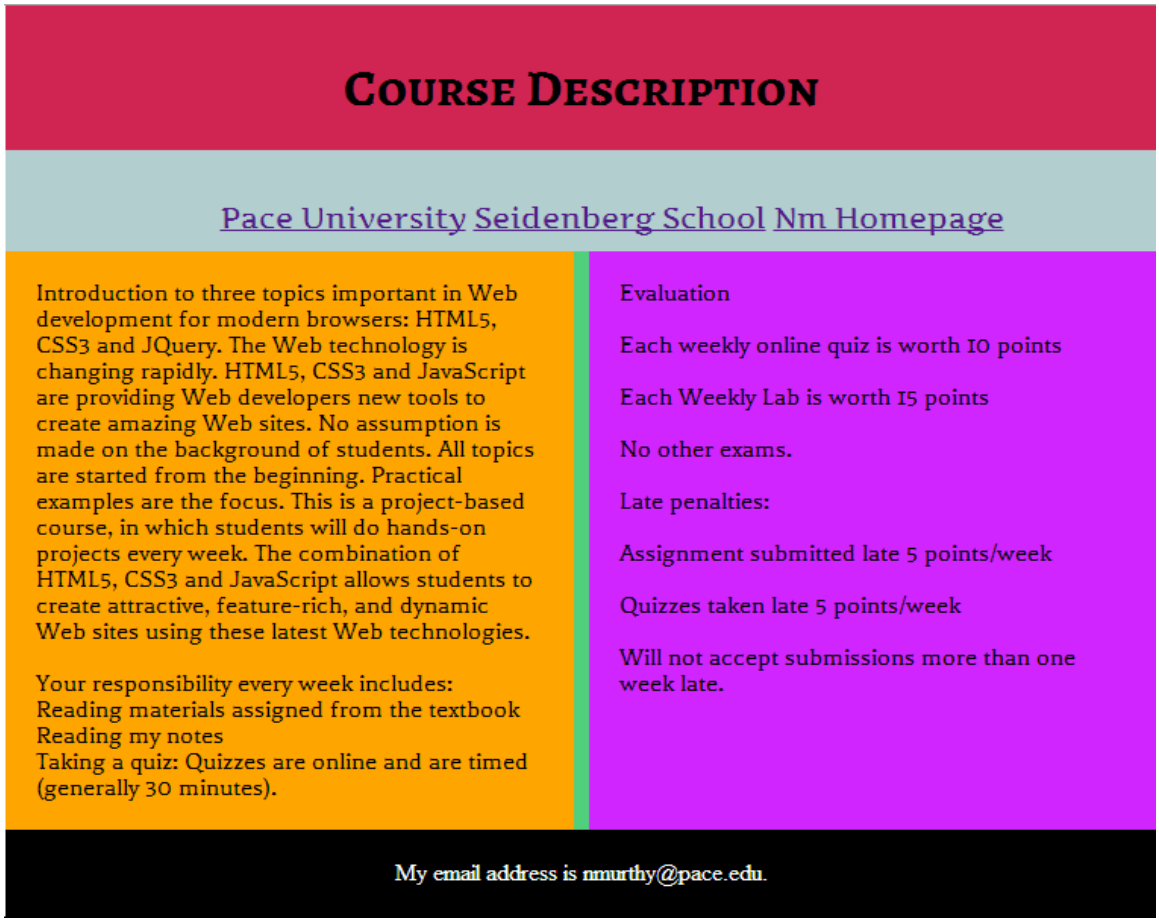
My email address is nmurthy@pace.edu.

Complete HTML page is here:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/twoColumnExample/twoColumnExampleHTML.pdf

Complete CSS page is here:

http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/twoColumnExample/twoColumnExampleCSS.pdf

Complete Web page in Chrome:



Try It:

http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/twoColumnExample/

23. Practice Examples

Before we continue with more CSS3 properties, let us do a couple of complete examples. We will develop a couple of Web pages, which use several CSS properties.

Before we start our examples, several review points:

1. The attributes name, class, and id

The **name** attribute is generally used in form elements such as input, textarea, select, and button elements. The value of the names attributes are sent to the server side program along with the data entered by the user. This happens when the user clicks the submit button. So the name attribute is not relevant in the context of CSS3.

Both id and class attributes are important in the context of CSS3. Both attributes are used frequently in HTML tags.

The value of an **id** attribute is unique in an HTML document. No two id value will be the same. Two HTML tags cannot have same the same value for the id attribute. The selector in CSS color definition corresponding to an id attribute is #idValue.

Example:
Consider the tag: ….
To refer to this particular tag in CSS color description you would use #one:
<style>
#one{ ……}
No other tag in the HTML document will have id value "one".

The value of a class attribute is not unique. Several tags in an HTML document can have the same value for class attribute. This helps us specify one set of CSS rules to several tags. The selector in CSS style definition corresponding to a class attribute is .classname.

**2. when to use and **
Use to wrap a big chunk of text – like a paragraph. Use selector div to specify CSS rules to apply to the entire paragraph. If you like to apply different CSS properties to a small piece of text within the paragraph use .

Example:

New York is the most populous city in the United States and the center of the New York Metropolitan Area, one of the most populous urban agglomerations in the world

Note: is a block element, is inline. It is illegal to place a block level element within an inline element, so:
New York is the most populous city in the United States and the center of the New York Metropolitan Area, one of the most populous urban agglomerations in the world

3. Use of <body> and together

Experts suggest that you should always start your body part of HTML document this way:

```
<body>
  <div id="container">
   …….
   …….
   …….
   </div>
</body>
```

Suggested CSS rules in these two:

```
body {
   background-color: #e1ddd9;
   font-size: 11px;
   font-family: Verdana, Arial, SunSans-Regular, Sans-Serif;
   color:#564b47;
}
#container {
   width: 1000px;
   border-style-bottom: 10px;
   border-style: 0 auto;
   background-color: #EBD3E0;
}
```

Study both HTML and CSS files in the following examples.

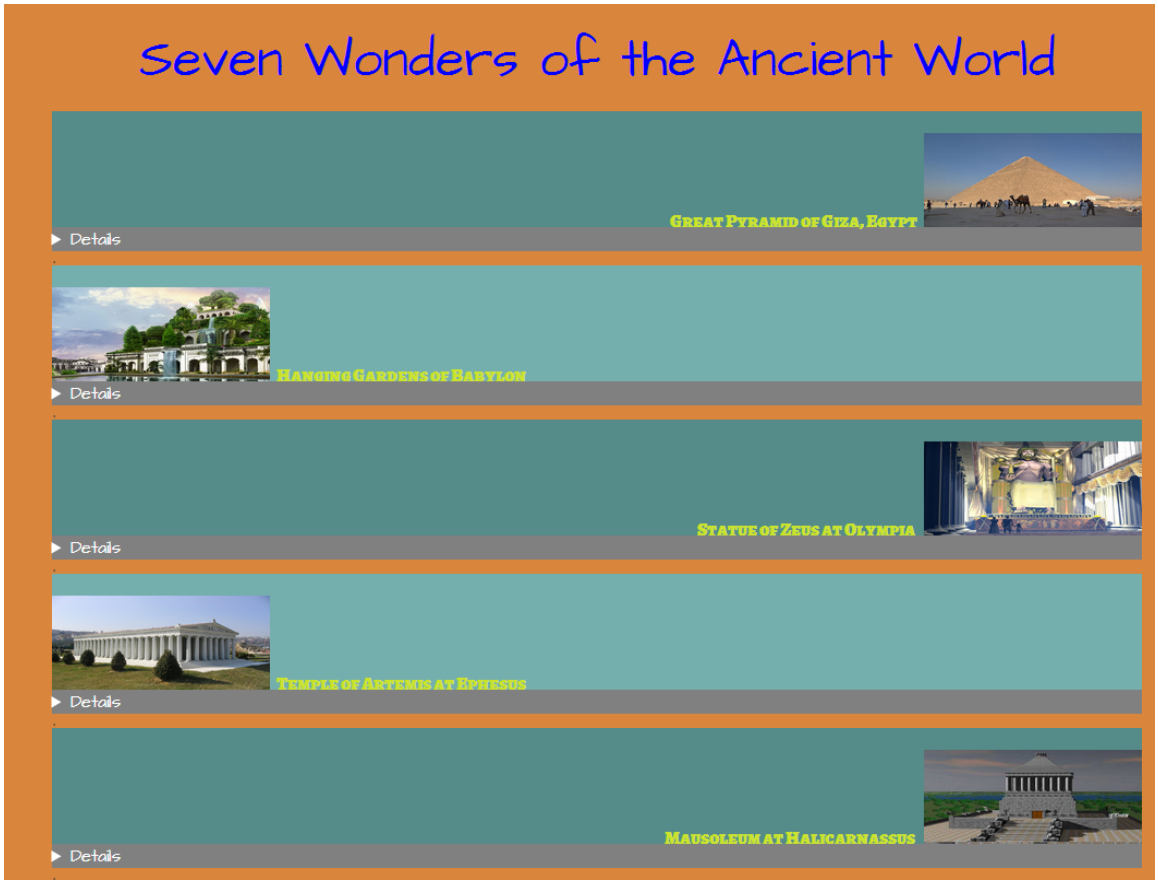
Practice Example 1: Seven Wonders of the Ancient World

This example displays the Seven Wonders of the Ancient World.

HTML file is here:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample1/PracticeExample1HTML.pdf

CSS file is here:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample1/PracticeExample1CSS.pdf

Chrome output:



Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample1

PRACTICE EXAMPLE 2: TOP MOVIES OF 2013

This example displays a list of Top Movies of 2013.
(It just shows the first page. I have not developed page for pther movies).

HTML file is here:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample2/Example2HTML.pdf

CSS file is here:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample2/Example2CSS.pdf

Chrome output:

Iron Man 3

When Tony Stark's world is torn apart by a formidable terrorist called the Mandarin, he starts an odyssey of rebuilding and retribution. (130 mins.) Iron Man 3 is the highest grossing movie of 2013 so far, and it is the only movie to cross over $1 billion at the worldwide box office. Iron Man 3 was also the most searched movie of 2013 on Bing. Following Iron Man 3 were Fast & Furious 6 and Despicable Me 2 as numbers two and three respectively. They were also the next two highest grossing movies at the worldwide box office after Iron Man 3, but the positions were reversed, as Despicable Me 2 finished ahead of Fast & Furious 6. In addition to Iron Man 3, The Wolverine finished as the fifth most searched movie and the Man of Steel finished as the tenth most searched movie. Did you see this mistake in the movie: When Tony is in his garage injecting his arm his chest piece is glowing brightly through his shirt, however, the very next shot when he is wiping his arm with the alcohol pad, his chest piece is suddenly turned off.

Director: Shane Black

Starring: Robert Downey, Jr., Gwyneth Paltrow, Don Cheadle, Guy Pearce, Rebecca Hall and Ben Kingsley

Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample2

PRACTICE EXAMPLE 3: IT614 COURSE DESCRIPTION

This example displays a a course description of this course IT614.

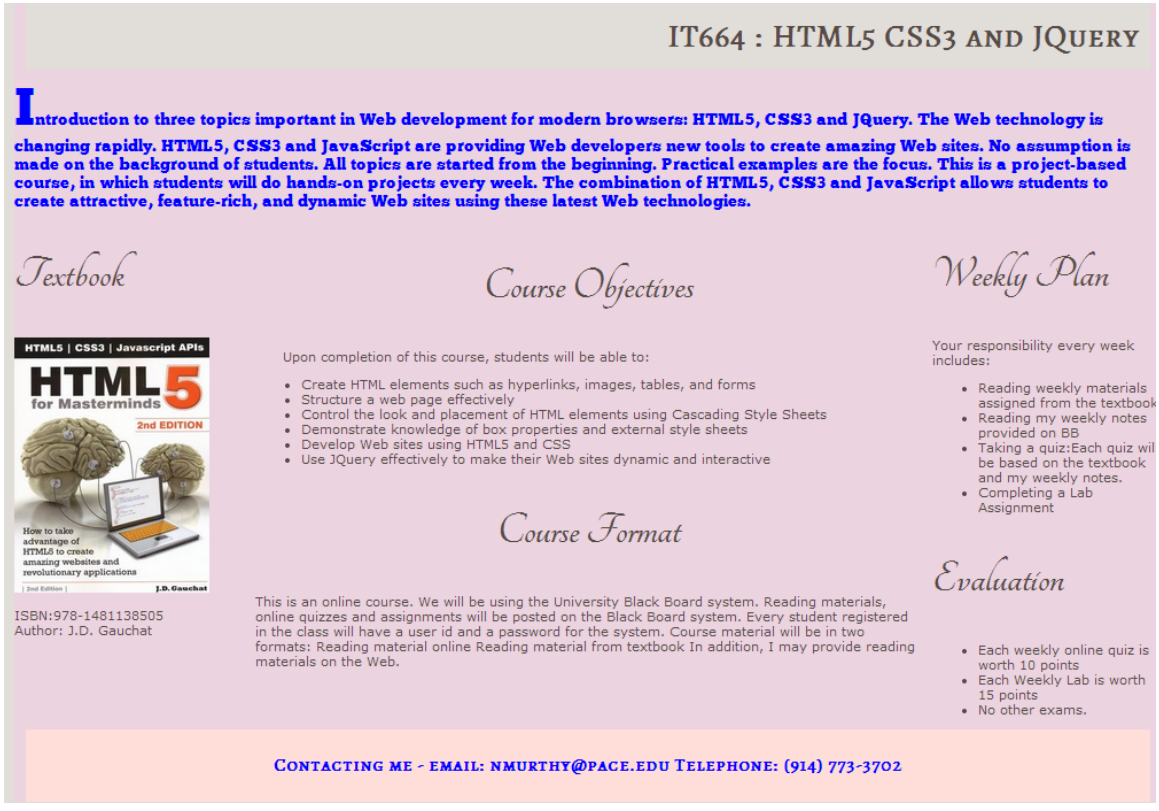
HTML file is here:

http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample3/PracticeExample3HTML.pdf

CSS file is here:

http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample3/PracticeExample3CSS.pdf

Chrome output:



Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/CSSExample3

---

24. CSS Flexbox

The CSS3 **Flexible Box**, or **flexbox**, is a new model for laying out HTML elements. Being a new technology, all Web browsers may not support all the features, yet. This new model promises to accommodate different screen sizes and different display devices, which is crucial for mobile devices. The flexbox model easier to use than the traditional box model. The best place to see the status of W3c recommendation is W3C Web site: http://www.w3.org/TR/css3-flexbox/. I have used this Web site in preparing these notes on flexbox model.

25. Flexbox vocabulary

- **New value for display property**

The flex model provides a new value for the CSS property **display** : **flex** and **inline-flex**.

- **Flex container and flex items**

An element with **display:flex** or **display:inline-flex** is called a flex container. The contents a flex container holds are called flex items.

- **Flex-flow directions**

A flex container has flow-direction as defined in the diagram below:

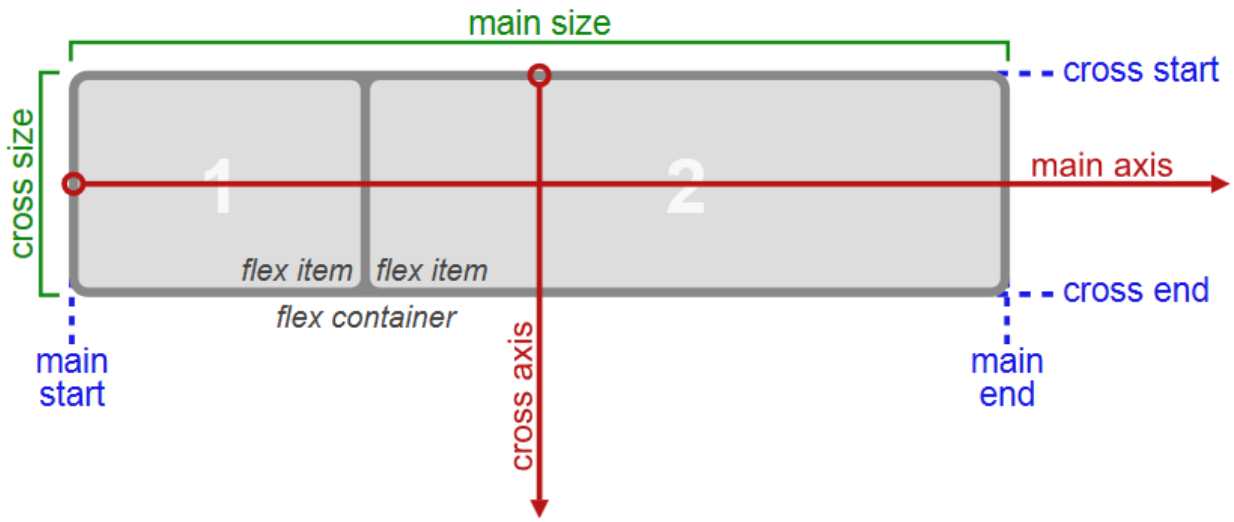


***Figure 2.** An illustration of the various directions and sizing terms as applied to a 'row' flex container.*

(http://www.w3.org/TR/css3-flexbox/).

Some definitions (see the diagram above):

- **main axis and main dimension**

The *main axis* of a flex container is the primary axis along which flex items are laid out. It extends in the *main dimension*.

- **main-start and main-end**

The flex items are placed within the container starting on the *main-start* side and going toward the *main-end* side.

- **main size and main size property**

A flex item's width or height, whichever is in the main dimension, is the item's *main size*. The flex item's *main size property* is either the '`width`' or '`height`' property, whichever is in the main dimension.

- **cross axis and cross dimension**

The axis perpendicular to the main axis is called the *cross axis*. It extends in the *cross dimension*.

- **cross-start and cross-end**

Flex lines are filled with items and placed into the container starting on the *cross-start* side of the flex container and going toward the *cross-end* side.

- **cross size and cross size property**

The width or height of a flex item, whichever is in the cross dimension, is the item's *cross size*. The *cross size property* is whichever of '`width`' or '`height`' that is in the cross dimension.

- **Designating an element as flexbox**

An element is made a flexbox by using the **display** property.

Syntax:

display : flex

This causes an element to generate a block-level flex container box.

display : inline-flex

This causes an element to generate an inline-level flex container box.

You can now format flex items in the flex container. Some of the previous box model properties are not applicable now. We will see details later.

The contents of a flex container consists of zero or more *flex items*: each child of a flex container becomes a flex item, and each contiguous run of text that is directly contained inside a flex container is wrapped in an anonymous flex item.

Example:

By default, in the older box model, elements are arranged one below the other. In a flexbox model they are arranged horizontally, one next to the other.

Consider the following HDML code with CSS:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <style>
#box1 {
  width: 150px;
  border: 5px inset #cc0000;
  color: white;
  padding: 5px;
  background-color: red;
}
#box2 {
  width: 150px;
  border: 5px inset #cc0000;
  color: white;
  padding: 5px;
  background-color: blue;
}
#box3 {
  width: 150px;
  border: 5px inset #cc0000;
```

```
  color: white;
  padding: 5px;
  background-color: green;
}
</style>
 </head>
 <body>
 <div id="container">
  <div id="box1">One</div>
  <div id="box2">Two</div>
  <div id="box3">Three</div>
</div>
 </body>
</html>
```

Output:

Now make the container a flexbox. Add the following code in CSS part:

```
#container {
    display : flex;
    background-color : gray;
}
```

The output now:

NOTE:

In our examples we have made the main container as a flexbox. You can make any element a flexbox.

26. Selected new CSS properties

We will explore several CSS properties which are applicable in flexbox model. We will use the three-box example in the following. I will show only the relevant CSS lines in the examples. Also, I have made the container background gray so that you can see the placement of the flex items in the flex container.

➢ **The CSS property flex-direction**

This property establishes the axis for placement of flex items in the container.

- **flex-direction : row;**

Flex items are placed left to right (default).

Example:

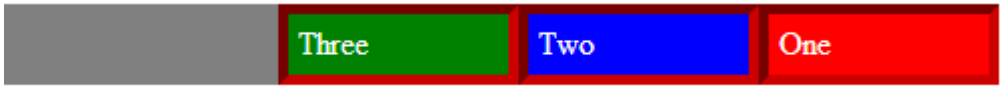
```
#container {
    display : flex;
    background-color : gray;
    flex-direction : row;
}
```

- **flex-direction : row-reverse;**

Flex items are placed right to left (from right margin).

Example:

```
#container {
    display : flex;
    background-color : gray;
    flex-direction : row-reverse;
}
```



- **flex-direction : column;**

Flex items are placed one below the other.

Example:

```
#container {
    display : flex;
    background-color : gray;
    flex-direction : column;
}
```

- **flex-direction : column-reverse;**

Flex items are placed one below the other.

Example:

```
#container {
    display : flex;
    background-color : gray;
    flex-direction : column-reverse;
}
```

➢ **The CSS property flex-wrap**

The CSS property **flex-wrap** defines whether the flex container is single-line or multi-line, and the direction of the cross-axis, which determines the direction new lines are stacked in.

- **flex-wrap : nowrap;**

Single line - No wrapping, left to right (default).

Example:

To understand wrapping, I have made the width of the container 300px.

```
#container {
    display : flex;
    background-color : gray;
    width : 300px;
    flex-wrap : nowrap;
}
```

NOTE:

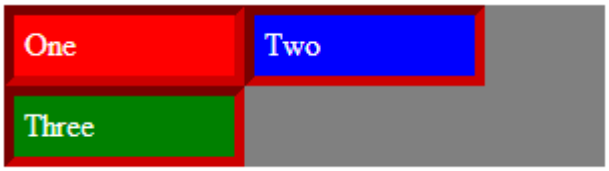
When you use **flex-wrap : nowrap;,** the flex items are laid out in a single line which may cause the flex container to overflow. That is what happened in this example. The flex container width is 300px, but the three flex items have taken more than 300 pixels.

- **flex-wrap : wrap;**

Multiple lines. Wrapping occurs if there is need.

Example:

```
#container {
    display : flex;
    background-color : gray;
    width : 300px;
    flex-wrap : wrap;
}
```



- **flex-wrap : wrap-reverse;**

Multiple lines. Wrapping occurs if there is need.

Example:

```
#container {
    display:flex;
    flex-wrap:wrap-reverse;
    width:500px;
}
```

➢ **The CSS property flex-flow**

The CSS **`flex-flow`** property is a shorthand property for **`flex-direction`** and **`flex-wrap`** individual properties.

The flex-flow property takes two values: flex-direction value and flex-wrap value.

EXAMPLE:

```
#container {
    display : flex;
    background-color : gray;
    width : 300px;
    flex-flow : row-reverse  wrap;
}
```

Other combinations of values are possible.

➢ **The CSS property justify-content**

The CSS `justify-content` property defines how a browser distributes available space between and around elements when aligning flex items in the main-axis of the current line. The alignment is done after the lengths and auto margins are applied, meaning that, if there is at least one flexible element, with `flex-grow` different than `0`, it will have no effect as there won't be any available space.

Values:
justify-content: flex-start
justify-content: flex-end
justify-content: center
justify-content: space-between
justify-content: space-around
justify-content: inherit

- **justify-content : flex-start;**

The flex items are packed starting from the main-start. Margins of the first flex item is flushed with the main-start edge of the line and each following flex item is flushed with the preceding. In other words, items are packed toward the start line (this is default).

Example:
I have made the container width 600px.

```
#container {
    display : flex;
    background-color : gray;
```

```
    width : 600px;
    justify-content : flex-start;
}
```

- **justify-content : flex-end;**

The flex items are packed starting from the main-end. The margin edge of the last flex item is flushed with the main-end edge of the line and each preceding flex item is flushed with the following. In other words, items are packed toward to end line.

Example:

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    justify-content : flex-end;
}
```

- **justify-content : center;**

The flex items are packed toward the center of the line. The flex items are flushed with each other and aligned in the center of the line. Space between the main-start edge of the line and first item and between main-end and the last item of the line is the same. In other words, items are centered along the line.

Example:

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    justify-content : center; }
```

- **justify-content : space-between;**

Flex items are evenly distributed along the line. The spacing is done such as the space between two adjacent items is the same. Main-start edge and main-end edge are flushed with respectively first and last flex item edges. In other words, items are evenly distributed in the line; first item is on the start line, last item on the end line.

Example:

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    justify-content : space-between;
}
```

- **justify-content : space-around;**

Flex items are evenly distributed so that the space between two adjacent items is the same. The empty space before the first and after the last items equals half of the space between two adjacent items. In other words, items are evenly distributed in the line with equal space around them.

Example:

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    justify-content : space-around;
}
```

➢ **The CSS property align-items**

The CSS `align-items` property aligns flex items of the current flex line the same way as `justify-content` but in the perpendicular direction.

Values:

align-items: flex-start

align-items: flex-end

align-items: center

align-items: baseline

align-items: stretch

align-items: inherit

- **align-items : flex-start;**

The cross-start margin edge of the flex item is flushed with the cross-start edge of the line.

Notice that we have included **flex-direction : column;** in the examples for #container..

Example:

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    flex-direction : column;
    align-items : flex-start;
}
```

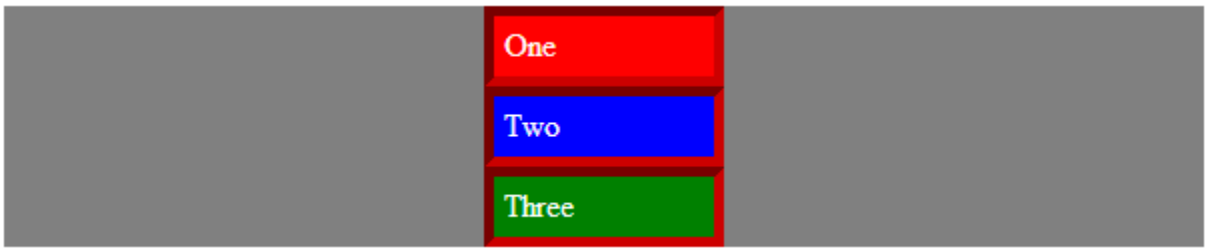
- **align-items : flex-end;**

The cross-end margin edge of the flex item is flushed with the cross-end edge of the line.

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    flex-direction : column;
    align-items : flex-end;
}
```

- **align-items : center;**

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    flex-direction : column;
    align-items : center;
}
```



- **align-items : baseline;**

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    flex-direction : column;
    align-items : baseline;
}
```

- **align-items : stretch;**

This is default.

```
#container {
    display : flex;
    background-color : gray;
    width : 600px;
    flex-direction : column;
    align-items : stretch;
}
```

➢ **The CSS property align-content**

Does not seem to work well in any of the browsers.

➢ **The CSS property order**

This is applied to a flex item (not a flex container).

The CSS property **order** specifies the order used to lay out flex items in their container. The value is an integer representing the rank.

EXAMPLE:

In the example, I have specified the following orders:

box One	order : 2;
box two	order : 3;
box three	order : 1;

Result:

- **The CSS property flex-grow**

This is applied to a flex item (not a flex container).

The value of flex-grow property is a number, indicating grow factor, the proportion by which the flex item will grow relative to the other flex items in the container.

Negative numbers are invalid.

EXAMPLE:

In the example, I have specified the following values for the three boxes:

box One	flex-grow:10;
box two	flex-grow:40;
box three	flex-grow:10;

Result:

- **The CSS property flex-shrink**

This is applied to a flex item (not a flex container).

The value of flex-shrink property is a number, indicating shrink factor, the proportion by which the flex item will shrink relative to the other flex items in the container.

Negative numbers are invalid.

EXAMPLE:

In the example, I have specified the following values for the three boxes:

box One	flex-shrink:10;
box two	flex-shrink:40;
box three	flex-shrink:10;

NOTE:

The property **flex-shrink** does not seem to work in any of the browsers I tried.

- **The CSS property flex-basis**

This is applied to a flex item (not to a flex container).

The **flex-basis** property specifies the initial main size of the flex item before free space is distributed. The initial main size is a flex item's **width** or **height** along the main (primary) axis. The initial value is **auto**, which retrieves the value of the main size of the flex container.

The value is a length in the usual length units.

EXAMPLE:

I have set box2 in our example, **flex-basis: 200px;**

Output:

- **The CSS property flex**

This is applied to a flex item (not to a flex container).

A shorthand property that specifies the parameter values of a flexible length, the positive and negative flexibility, and the preferred size specified by the **flex-grow**, **flex-shrink**, and **flex-basis** properties.

The property takes a set of three vaues:

flex : *value1 value2 value3*;

value1 is flex-grow value
value2 is flex-shrink value
value2 is flex-basis value

- **The CSS property align-self**

Specifies the alignment value (perpendicular to the layout axis defined by the **flex-direction** property) of flex items of the flex container.
Values:
auto
Initial value. If the flex item has a parent flex container, **align-self** is equal to the **align-items** property of the parent. Otherwise, it's equal to stretch.

flex-start
If the flex container has a computed value for **flex-direction** of "row" or "column", the leading edge (or baseline) of each flex item is aligned with the leading edge of the flex container. Any remaining space, perpendicular to the layout axis, is placed after the trailing edge of each flex item.
If the flex container has a computed value for **flex-direction** of "row-reverse" or "column-reverse", the trailing edge (or baseline) of each flex item is aligned with the trailing edge of the flex container. Any remaining space, perpendicular to the layout axis, is placed before the leading edge of each flex item.

flex-end
If the flex container has a computed value for **flex-direction** of "row" or "column", the trailing edge of each flex item is aligned with the trailing edge of the flex container. Any remaining space, perpendicular to the layout axis, is placed before the leading edge of each flex item.
If the flex container has a computed value for **flex-direction** of "row-reverse" or "column-reverse", the leading edge of each flex item is aligned with the leading edge of the flex container. Any remaining space, perpendicular to the layout axis, is placed after the trailing edge of each flex item.

center

Each flex item is centered between the leading and trailing edges of the flex container. Any remaining space, perpendicular to the layout axis, is evenly distributed before and after flex item.

baseline

The baselines (leading edge or trailing edge depending on the **flex-direction** property) of all flex items are aligned with each other.
The flex item that occupies the most space, perpendicular to the layout axis, follows the flex-start rule. The baselines of all remaining flex items are then aligned with the baseline of this element.

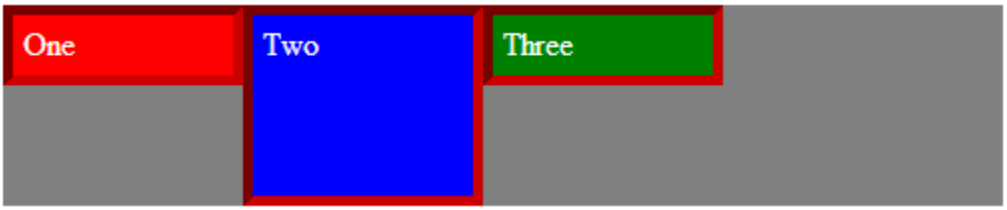
stretch

Each flex item is stretched to completely fill the space that is available perpendicular to the layout axis. If set, height or width related properties for a flex item takes precedence and layout follows the flex-start rule.

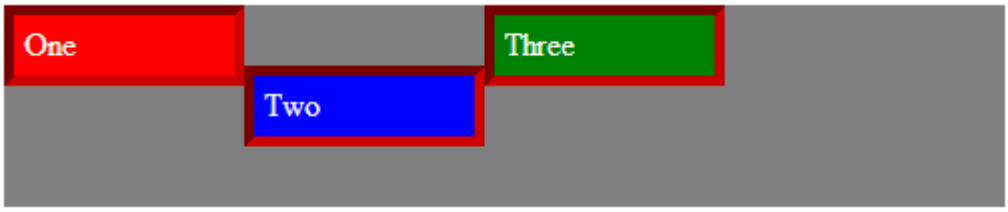
EXAMPLE:

```
#container {
    display : flex;
    background-color : gray;
    width : 500px;
    height : 100px;
    align-items: flex-start;
}
```

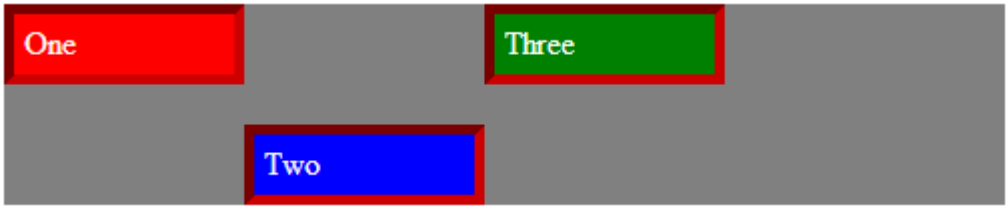
and in Box2, **align-self: stretch;**



In box2, **align-self: center;**



In box2, **align-self: flex-end;**



---

27. A complete example using flexbox properties

This example is a simple Web page for a travel site. Please study both the HTML code and CSS style sheet.

I have posted the HTML and CSS files here:
HTML:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/FlexExample/flexExampleHTML.pdf

CSS:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/FlexExample/flexExampleCSS.pdf

Chrome output:

Try it:

http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter2/FlexExample/

Notes:

1. Please remember that names of files (such as image files) you use in your HTML documents are case sensitive. cat.jpg and Cat.jpg are not the same. They may work fine on your Windows machine, but when you upload to Vulcan, it may not work. This is because Vulcan is a Linux machine where everything is strictly case sensitive.
2. After you are done writing your HTML5 file, you can validate it using one of many validators available online. One good HTML5 validator is here: http://validator.w3.org/#validate_by_input. Copy and paste your HTML code in the direct Input box and click Check. If I shows Errors, fix the mistakes. If it says, warnings – study the warnings and you need not worry too much about them. In most cases the warnings are new tags in HTML5.
3. You can also validate your CSS code here: http://jigsaw.w3.org/css-validator/#validate_by_input.
4. You can even "beautify" your HTML and CSS code here: http://www.dirtymarkup.com/. Play with it, it is interesting.
