



مجمع فنی تهران

CSS3

Tehran Institute of Technology

Zahra Mansoori

z.mansoori@gmail.com

<http://tarsimm.com>

Contents

1. Introduction	6
1.1. HTML Syntax	6
1.1.1. HTML class Attribute	6
1.1.2. HTML id Attribute	7
1.2. CSS – SYNTAX	7
1.3. The Type Selectors	8
1.3.1. The Universal Selectors	8
1.3.2. The Descendant Selectors	8
1.3.3. The Class Selectors	8
1.3.4. The ID Selectors	9
1.3.5. The Child Selectors	10
1.3.6. The Attribute Selectors	10
1.3.7. Multiple Style Rules	11
1.3.8. Grouping Selectors	12
2. CSS – INCLUSION	13
2.1. Embedded CSS - The <style> Element	13
2.2. Inline CSS - The style Attribute	13
2.3. External CSS - The <link> Element	14
2.4. Imported CSS - @import Rule	15
2.5. CSS Rules Overriding	16
2.6. Comment tags	16
3. CSS – MEASUREMENT UNITS	17
4. CSS Classes and IDs	18
4.1. CSS Classes	18
4.2. CSS IDs	19
4.2.1. Divisions	20
4.2.2. Spans	20
5. CSS syntaxes	22
5.1. Margins, Paddings, Borders	22
5.1.1. Padding	22
5.1.2. Border	23
5.1.3. Margin	23

5.2. CSS Margins.....	24
5.3. CSS Padding.....	25
5.4. Border	27
5.4.1. Border Color.....	27
5.4.2. Border Style	28
5.4.3. Border Width	28
5.5. Border Bottom	29
5.5.1. Border Bottom Color.....	29
5.5.2. Border Bottom Style	29
5.5.3. Border Bottom Width	30
5.6. Border Left	30
5.6.1. Border Left Color.....	30
5.6.2 Border Left Style	30
5.6.3. Border Left Width	30
5.7. Border Right.....	31
5.7.1. Border Right Color	31
Border Right Style	31
5.7.2. Border Right Width	31
5.8. Border Top	32
5.8.1. Border Top Color.....	32
5.8.2. Border Top Style	32
5.8.3. Border Top Width	32
6. CSS Text Properties.....	34
6.1. Color.....	34
6.2. Letter Spacing	34
6.3. Text Align	34
6.5. Text Decoration.....	35
6.6. Text Indent.....	36
6.7. Text Transform.....	36
6.8. White Space	36
6.9. Word Spacing.....	37
7. CSS Font Properties.....	38
7.1. Font –Family.....	38

7.2. Font Size	38
7.3. Font Style	39
7.4. Font Variant	39
7.5. Font Weight	40
7.6. Font	40
8. CSS Anchors, Links and Pseudo Classes	41
8.1. Pseudo Classes	42
8.2. All Psoudo-classes	43
9. CSS Backgrounds	46
9.1. Background Attachment	46
9.2. Background Color	46
9.3. Background Image	46
9.4. Background	47
9.5 Background Repeat	47
9.6. ackground Origin	48
9.7. Background Size	48
9.8. Background	49
10. CSS Ordered & Unordered Lists	50
10.1. List Style Image	50
10.2. List Style Position	50
10.3. List Style Type	50
10.5. List Style	51
11. CSS Width and Height Properties	52
11.1. Height	52
11.2. Line Height	52
11.3. Max Height	52
11.4. Min Height	53
11.5. Width	53
11.6. Max Width	53
11.7. Min Width	53
12. CSS Positioning	55
12.1. Position	55
13. CSS Classification	57

13.1. Float	57
13.2. Display.....	57
13.3. Clear	58
13.4. Clip	58
13.5. Cursor.....	59
13.6. Overflow	59
13.7. Visibility.....	60
13.8. Z-Index	60
14. Visibility Problem	61
14.1. Visibility property.....	61
14.2. Display property.....	61
14.3. Opacity Property	61
15. CSS Pseudo Elements	62
15.1. The Syntax.....	62
15.2. First Line.....	62
15.3. First-line with class.....	62
15.4. First Letter.....	63
15.5. First-letter with class.....	63
15.6. ::before	64
15.7. ::after	64
Hint:	64
15.8. ::selection.....	64
15.9. ::placeholder	65
16. CSS Transition and Transform.....	66
16.1. CSS transitions: an introduction	66
Hint: Transition Shorthand	67
16.1.1. transition-property (required):	67
16.1.2. transition-duration (required):	67
Hint: Shorthand example for transition-duration.....	68
16.1.3. transition-timing (optional):	68
16.1.4. transition-delay (optional):	68
Hint: Shorthand example for transition-delay	68
16.2. CSS transforms: an introduction	69

16.2.1. scale	69
16.2.2. rotate	70
16.2.3. translate	70
HTML :	71
16.2.4. skew	71
Hint	72
16.2.5. transform-origin.....	72
16.3. Combining transforms	73
17. CSS Animation.....	74
17.1. The Building Blocks of Animations.....	74
17.1.1. Keyframes	74
17.1.2. Animation Properties.....	75
17.2. Additional Animation Properties	76
17.2.1. Animation-timing-function	76
Hint	77
17.2.2. Animation-Delay	77
Hint: Animation shorthand syntax (recommended):.....	77
17.2.3. Animation-iteration-count.....	77
Hint	77
17.2.4. Animation-direction.....	78
Hint	78
17.2.5. Animation-fill-mode.....	78
Hint	79
17.2.6. Animation-play-state	79
17.2.7. Multiple Animations	79
17.3. Note About Prefixes.....	79

1. Introduction

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects.

CSS is easy to learn and understand but it provides a powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.



1.1. HTML Syntax

It is needed to clarify two syntax of HTML, which are used to define **Class** and **ID**:

1.1.1. HTML class Attribute

The class attribute specifies one or more class names for an element to point to the classes in a style sheet.

```
<body>

    <h1 class="intro">Header 1</h1>
    <p>A paragraph.</p>
    <p class="important">Note that this is an important paragraph. :)</p>

</body>
```

1.1.2. HTML id Attribute

The id attribute specifies a unique id for an HTML element (the value must be unique within the HTML document). The id attribute is most used to point to a style in a style sheet.

```
<body>

    <h1 id="myHeader">Hello World!</h1>

</body>
```

1.2. CSS – SYNTAX

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

- **Selector:** A selector is an HTML tag at which a style will be applied. This could be any tag like `<h1>` or `<table>` etc.
- **Property:** A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be `color`, `border`, etc.
- **Value:** Values are assigned to properties. For example, `color` property can have the value either `red` or `#F1F1F1` etc.

You can put CSS Style Rule Syntax as follows:

```
selector { property: value }
```

You can define a table border as follows:

```
table{ border :1px solid #C00; }
```

Here `table` is a selector and `border` is a property and the given value `1px solid #C00` is the value of that property.

1.3. The Type Selectors

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings:

```
h1 {  
    color: #36CFFF;  
}
```

1.3.1. The Universal Selectors

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type:

```
* {  
    color: #000000;  
}
```

This rule renders the content of every element in our document in black.

1.3.2. The Descendant Selectors

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, the style rule will apply to `` element only when it lies inside the `` tag.

```
ul em {  
    color: #000000;  
}
```

1.3.3. The Class Selectors

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {
```

```
color: #000000;

}
```

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular. For example:

```
h1.black {

    color: #000000;

}
```

This rule renders the content in black for only <h1> elements with class attribute set to black.

You can apply more than one class selectors to a given element. Consider the following example:

```
<p class="center bold">

    This para will be styled by the classes center and bold.

</p>
```

1.3.4. The ID Selectors

You can define style rules based on the **id** attribute of the elements. All the elements having that **id** will be formatted according to the defined rule.

```
#black {

    color: #000000;

}
```

This rule renders the content in black for every element with **id** attribute set to black in our document. You can make it a bit more particular. For example:

```
h1#black {

    color: #000000;

}
```

This rule renders the content in black for only `<h1>` elements with `id` attribute set to black.

The true power of `id` selectors is when they are used as the foundation for descendant selectors. For example:

```
#black h2 {  
    color: #000000;  
}
```

In this example, all level 2 headings will be displayed in black color when those headings will lie within tags having id attribute set to black.

1.3.5. The Child Selectors

You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example:

```
body > p {  
    color: #000000;  
}
```

This rule will render all the paragraphs in black if they are a direct child of the `<body>` element. Other paragraphs put inside other elements like `<div>` or `<td>` would not have any effect of this rule.

1.3.6. The Attribute Selectors

You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of *text*:

```
input[type="text"]{  
    color: #000000;  
}
```

The advantage to this method is that the `<input type="submit" />` element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.

- **p[lang]** - Selects all paragraph elements with a *lang* attribute.
- **p[lang="fr"]** - Selects all paragraph elements whose *lang* attribute has a value of exactly "fr".
- **p[lang~="fr"]** - Selects all paragraph elements whose *lang* attribute contains the word "fr".
- **p[lang|="en"]** - Selects all paragraph elements whose *lang* attribute contains values that are exactly "en", or begin with "en-".

1.3.7. Multiple Style Rules

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example:

```
h1 {  
    color: #36C;  
    font-weight: normal;  
    letter-spacing: .4em;  
    margin-bottom: 1em;  
    text-transform: lowercase;  
}
```

Here all the property and value pairs are separated by a **semicolon (;)**. You can keep them in a single line or multiple lines. For better readability, we keep them in separate lines.

For a while, don't bother about the properties mentioned in the above block. These properties will be explained in the coming chapters and you can find the complete detail about properties in CSS References.

1.3.8. Grouping Selectors

You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example:

```
h1, h2, h3 {  
    color: #36C;  
    font-weight: normal;  
    letter-spacing: .4em;  
    margin-bottom: 1em;  
    text-transform: lowercase;  
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

You can combine the various class selectors together as shown below:

```
#content, #footer, #supplement {  
    position: absolute;  
    left: 510px;  
    width: 200px;  
}
```

2. CSS – INCLUSION

There are four ways to associate styles with your HTML document. Most commonly used methods are inline CSS and External CSS.

2.1. Embedded CSS - The <style> Element

You can put your CSS rules into an HTML document using the <style> element. This tag is placed inside the <head>...</head> tags. Rules defined using this syntax will be applied to all the elements available in the document. Here is the generic syntax:

```
<head>
<style type="text/css" media="...">
    Style Rules
    .....
</style>
</head>
```

Following is an example of embed CSS based on the above syntax:

```
<head>
<style type="text/css" media="all">
    h1{
        color: #36C;
    }
</style>
</head>
```

2.2. Inline CSS - The style Attribute

You can use *style* attribute of any HTML element to define style rules. These rules will be applied to that element only. Here is the generic syntax:

```
<element style="...style rules....">
```

Following is the example of inline CSS based on the above syntax:

```
<h1 style="color:#36C;"> This is inline CSS </h1>
```

2.3. External CSS - The <link> Element

The <link> element can be used to include an external stylesheet file in your HTML document.

An external style sheet is a separate text file with .css extension. You define all the Style rules within this text file and then you can include this file in any HTML document using <link> element.

Here is the generic syntax of including external CSS file:

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

Consider a simple style sheet file with a name mystyle.css having the following rules:

mystyle.css

```
h1, h2, h3 {
color: #36C;
font-weight: normal;
letter-spacing: .4em;
margin-bottom: 1em;
text-transform: lowercase;
}
```

Now you can include this file mystyle.css in any HTML document as follows:

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

2.4. Imported CSS - @import Rule

@import is used to import an external stylesheet in a manner similar to the `<link>` element. Here is the generic syntax of **@import** rule.

```
<head>
<style type="text/css">
@import "URL";
</style>
</head>
```

Here URL is the URL of the style sheet file having style rules. You can use another syntax as well:

```
<head>
<style type="text/css">
@import url("URL");
</style>
</head>
```

Following is the example showing you how to import a style sheet file into an HTML document:

```
<head>
<style type="text/css">
@import "mystyle.css";
```



```
</style>
</head>
```

2.5. CSS Rules Overriding

We have discussed four ways to include style sheet rules in an HTML document. Here is the rule to override any Style Sheet Rule.

- Any inline style sheet takes the **highest priority**. So, it will override any rule defined in `<style>...</style>` tags or the rules defined in any external style sheet file.
- Any rule defined in `<style>...</style>` tags will override the rules defined in any external style sheet file.
- Any rule defined in the external style sheet file takes the **lowest priority**, and the rules defined in this file will be applied only when the above two rules are not applicable.

2.6. Comment tags

Comments can be used to explain why you added certain selectors within your css file. To help others who may see your file or to help you remember what you we're thinking at a later date. You can add comments that will be ignored by browsers in the following manner.

```
/* This is a comment */
```

3. CSS – MEASUREMENT UNITS

Before we start the actual exercise, we would like to give a brief idea about the CSS Measurement Units. CSS supports a number of measurements including absolute units such as inches, centimeters, points, and so on, as well as relative measures such as percentages and em units. You need these values while specifying various measurements in your Style rules e.g. `border="1px solid red"`.

Unit	Description	Example
%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	p {font-size: 16pt; line-height: 125%;}
cm	Defines a measurement in centimeters.	div {margin-bottom: 2cm;}
em	A relative measurement for the height of a font in em spaces. Because an em unit is equivalent to the size of a given font, if you assign a font to 12pt, each "em" unit would be 12pt; thus, 2em would be 24pt.	p {letter-spacing: 7em;}
ex	This value defines a measurement relative to a font's x-height. The x-height is determined by the height of the font's lowercase letter x.	p {font-size: 24pt; line-height: 3ex;}
in	Defines a measurement in inches.	p {word-spacing: .15in;}
mm	Defines a measurement in millimeters.	p {word-spacing: 15mm;}
pc	Defines a measurement in picas. A pica is equivalent to 12 points; thus, there are 6 picas per inch.	p {font-size: 20pc;}
pt	Defines a measurement in points. A point is defined as 1/72nd of an inch.	body {font-size: 18pt;}
px	Defines a measurement in screen pixels.	p {padding: 25px;}

4. CSS Classes and IDs

4.1. CSS Classes

The class selector allows you to style items within the same HTML element differently. You can use the same class selector repeatedly within an HTML file.

To put it more simply, the sentence you are reading is defined in my CSS file with the following.

```
p {  
    font-size: small;  
    color: #333333  
}
```

I wanted to change the word "sentence" to green bold text, while leaving the rest of the sentence untouched. I would do the following to my HTML file.

```
<p>  
To put it more simply, this  
    <span class="greenboldtext">sentence</span>  
you are reading is styled in my CSS file by the following.  
</p>
```

Then in my CSS file, I would add this style selector:

```
.greenboldtext {  
    font-size: small;  
    color: #008080;  
    font-weight: bold;  
}
```

The result would look like the following:

To put it more simply, this **sentence** you are reading is styled in my CSS file by the following.

4.2. CSS IDs

IDs are similar to **classes**, except once a specific id has been declared it cannot be used again within the same HTML file.

Generally, IDs are used to style the layout elements of a page that will only be needed once, whereas **classes are used** to style text and such that may be declared multiple times.

The main container for this page is defined by the following:

```
<div id="container">
  Everything within my document is inside this division.
</div>
```

Id selector is chosen for the "container" division over a class, because It is only need to use it one time within this file.

Then in my CSS file, I have the following:

```
#container{
  width: 80%;
  margin: auto;
  padding: 20px;
  border: 1px solid #666;
  background: #ffffff;
}
```

You will notice that the id selector begins with a **#** number sign instead of a **.** period, as the class selector does.

Here we make some examples in order you to get familiar with these concepts:

4.2.1. Divisions

Divisions are a block level HTML element used to define sections of an HTML file. A division can contain all the parts that make up your website. Including additional divisions, spans, images, text and so on.

You define a division within an HTML file by placing the following between the `<body></body>` tags:

```
<div>
    Site contents go here
</div>
```

Though most likely you will want to add some style to it. You can do that in the following fashion:

```
<div id="container">
    Site contents go here
</div>
```

The CSS file contains this:

```
#container{
    width: 70%;
    margin: auto;
    padding: 20px;
    border: 1px solid #666;
    background: #ffffff;
}
```

4.2.2. Spans

Spans are very similar to divisions except they are an inline element versus a block level element. No line break is created when a span is declared.

You can use the span tag to style certain areas of text, as shown in the following:

```
<span class="italic">This text is italic</span>
```

Then in my CSS file:

```
.italic{  
    font-style: italic;  
}
```

The result is:

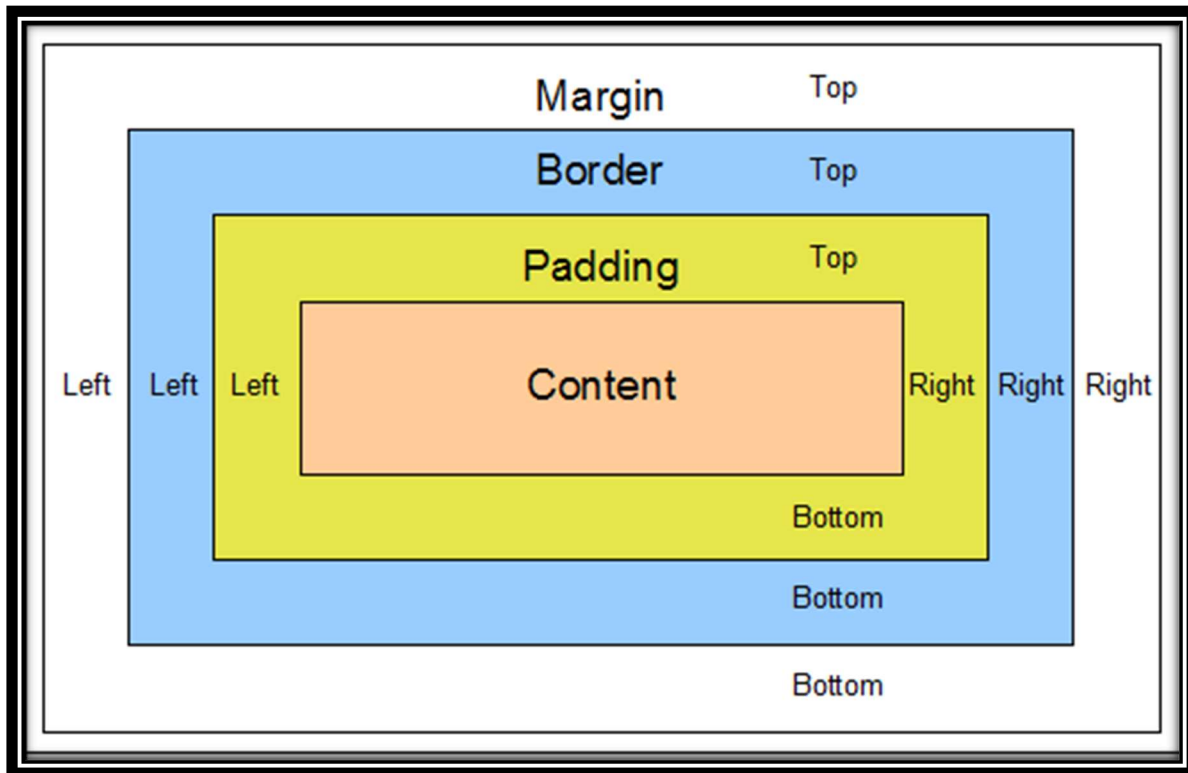
This text is italic

5. CSS syntaxes

5.1. Margins, Paddings, Borders

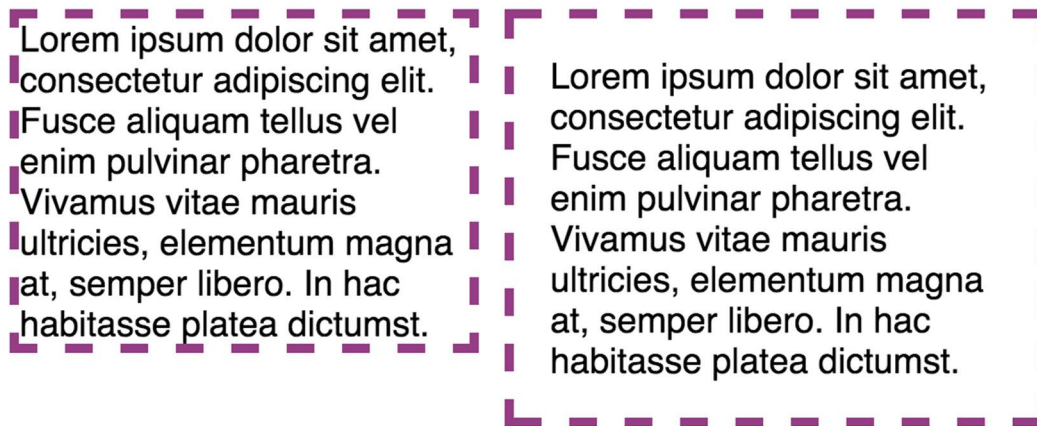
Every HTML element you see on the page is treated as if it lives in its own box by the CSS box model, which the diagram below illustrates.

Inside every element, you can have content such as a title or icon and it is sized just enough to hold that content. To adjust the size you can set the dimensions by using the height and width properties. You can also control the appearance of the box by using the padding, border, and margin properties. Whenever the padding, border, and margin properties are used, they will increase the width and height of the box.



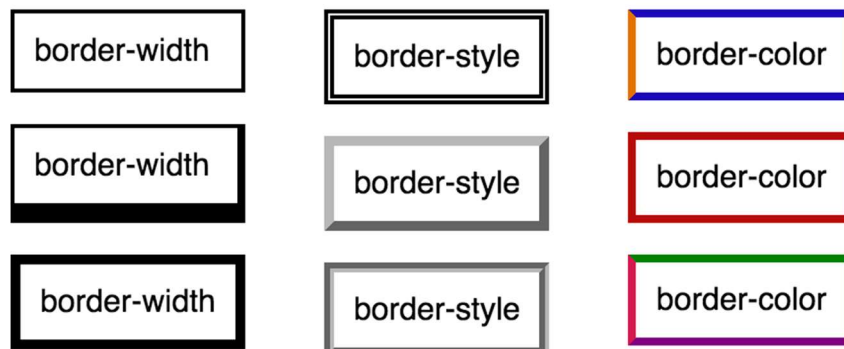
5.1.1. Padding

The space that surrounds the area between the content and border is called padding. You can choose to add padding to every side, one side, a few sides, or specify no amount at all. By specifying how much space should appear between the content and border you can make it easier for people to read the element's contents such as a paragraph.



5.1.2. Border

Around every element's box is a border separating it from other elements on the page whether you see it or not. The border is the element's edge that surrounds the content and padding. You can control the border's width, style, and color of any side of the element but you must set a value for each or the border will not appear.



5.1.3. Margin

The final way you can change the appearance of an element is by using the margin property. Margin is outside the edge of an element and it clears an area outside of the border to create gaps between elements.

Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Fusce aliquam tellus vel
enim pulvinar pharetra.

Vivamus vitae mauris
ultrices, elementum magna
at, semper libero. In hac
habitasse platea dictumst.

Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Fusce aliquam tellus vel enim
pulvinar pharetra.

Vivamus vitae mauris ultricies,
elementum magna at, semper
libero. In hac habitasse platea
dictumst.

5.2. CSS Margins

As you may have guessed, the margin property declares the margin between an HTML element and the elements around it. The margin property can be set for the top, left, right and bottom of an element.

```
margin-top: length percentage or auto;  
margin-left: length percentage or auto;  
margin-right: length percentage or auto;  
margin-bottom: length percentage or auto;
```

As you can also see in the above example, you have three choices of values for the margin property

- length
- percentage
- auto

You can also declare all the margins of an element in a single property as follows:

```
margin: 10px 10px 10px 10px;
```

If you declare all four values as I have above, the order is as follows:

1. top
2. right
3. bottom
4. left

If only one value is declared, it sets the margin on all sides.

```
margin: 10px;
```

If you only declare two or three values, the undeclared values are taken from the opposing side.

```
margin: 10px 10px; /* 2 values */  
margin: 10px 10px 10px; /* 3 values */
```

You can set the margin property to negative values. If you do not declare the margin value of an element, the margin is zero.

```
margin: -10px;
```

Elements like paragraphs have default margins in some browsers, to combat this set the margin to zero.

```
p {margin: 0;}
```

Note: You do not have to add px (pixels) or whatever units you use, if the value is zero.

You can see in the example below, the elements for this site are set to be 20px (pixels) from the body.

```
body{  
    margin: 20px;  
    background: #eeeeee;  
    font-size: small;  
    font-family: Tahoma, Arial, "Trebuchet MS", Helvetica, sansserif;  
    text-align: left;  
}
```

5.3. CSS Padding

Padding is the distance between the border of an HTML element and the content within it.

Most of the rules for **margins** also apply to padding, except there is no "auto" value, and negative values cannot be declared for padding.

```
padding-top: length percentage;  
padding-left: length percentage;  
padding-right: length percentage;  
padding-bottom: length percentage;
```

As you can also see in the above example, you have two choices of values for the padding property:

- Length
- Percentage

You can also declare all the padding of an element in a single property as follows:

```
padding: 10px 10px 10px 10px;
```

If you declare all four values as I have above, the order is as follows:

1. top
2. right
3. bottom
4. left

If only one value is declared, it sets the padding on all sides:

```
padding: 10px;
```

If you only declare two or three values, the undeclared values are taken from the opposing side.

```
padding: 10px 10px; /* 2 values */  
padding: 10px 10px 10px; /* 3 values */
```

If you do not declare the padding value of an element, the padding is zero.

Note: You do not have to add px (pixels) or whatever **units** you use, if the value is 0.

You can see in the example below, the main container for this site has 30px (pixels) of padding between the border and the text.

```
#container{
    width: 70%;
    margin: auto;
    padding: 30px;
    border: 1px solid #666;
    background: #ffffff;
}
```

5.4. Border

You can set the color, style and width of the borders around an element in one declaration by using the border property.

```
border: 1px solid #333333;
```

Values:

- color
- style
- width

Or you can set each property individually.

5.4.1. Border Color

You can set the color of a border independently with the border-color property.

```
border-color: value;
```

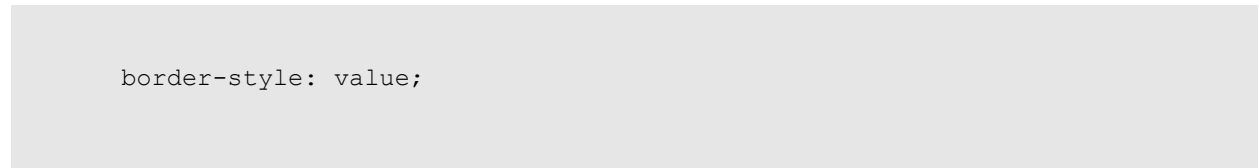


Values:

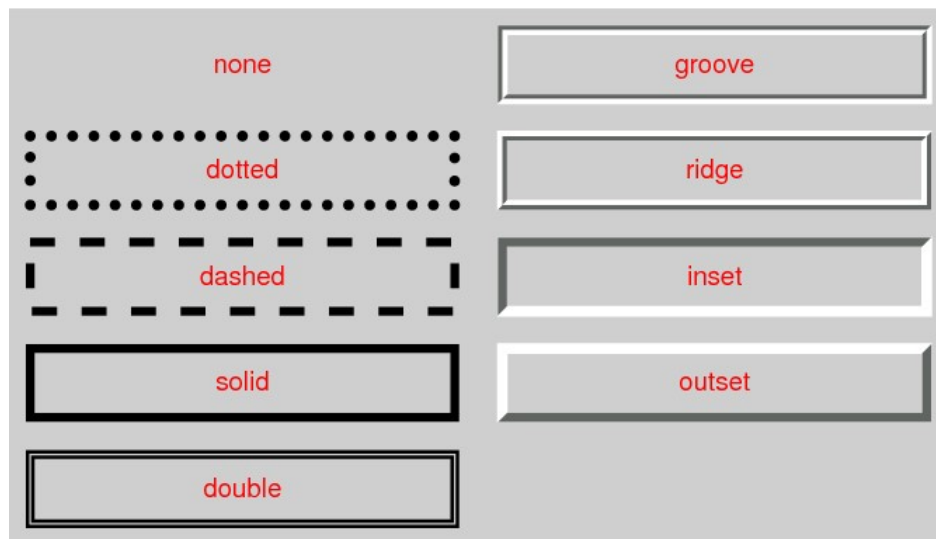
- color name
- hexadecimal number
- RGB color code
- Transparent

5.4.2. Border Style

You can set the style of a border independently with the border-style property.

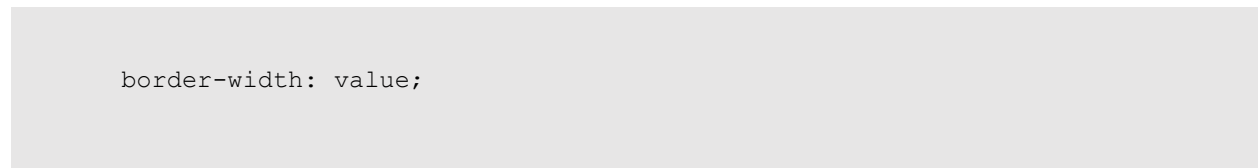


Values:



5.4.3. Border Width

You can set the width of a border independently with the border-width property.



Values:

- Length

- Thin
- Medium
- Thick

Or you can set the elements for each borders side individually.

5.5. Border Bottom

You can set the color, style and width of the bottom border around an element in one declaration with the border-bottom property.

```
border-bottom: 1px solid #333333;
```

Values:

- color
- style
- width

Alternatively, you can set each value individually.

5.5.1. Border Bottom Color

You can set the color of the bottom border around an element with the border-bottom-color property.

```
border-bottom-color: value;
```

5.5.2. Border Bottom Style

You can set the style of the bottom border around an element with the border-bottom-style property.

```
border-bottom-style: value;
```

5.5.3. Border Bottom Width

You can set the width of the bottom border around an element with the `border-bottom-width` property.

```
border-bottom-width: value;
```

5.6. Border Left

You can set the color, style and width of the left border around an element with the `border-left` property.

```
border-left: 1px solid #333333;
```

Values:

- color
- style
- width

On the other hand, you can set each value individually.

5.6.1. Border Left Color

You can set the color of the left border around an element with the `border-left-color` property.

```
border-left-color: value;
```

5.6.2 Border Left Style

You can set the style of the left border around an element with the `border-left-style` property.

```
border-left-style: value;
```

5.6.3. Border Left Width

You can set the width of the left border around an element with the `border-left-width` property.

```
border-left-width: value;
```

5.7. Border Right

You can set the color, style and width of the right border around an element in one declaration with the `border-right` property.

```
border-right: 1px solid #333333;
```

Values:

- color
- style
- width

On the other hand, you can set each value individually.

5.7.1. Border Right Color

You can set the color of the right border around an element with the `border-right-color` property.

```
border-right-color: value;
```

Border Right Style

You can set the style of the right border around an element with the `border-right-style` property.

```
border-right-style: value;
```

5.7.2. Border Right Width

You can set the width of the right border around an element with the `border-right-width` property.


```
border-right-width: value;
```

5.8. Border Top

You can set the color, style and width of the top border around an element in one declaration with the border-top property.

```
border-top: 1px solid #333333;
```

Values:

- color
- style
- width

On the other hand, you can set each value individually.

5.8.1. Border Top Color

You can set the color of the top border around an element with the border-top-color property.

```
border-top-color: value;
```

5.8.2. Border Top Style

You can set the style of the top border around an element with the border-top-style property.

```
border-top-style: value;
```

5.8.3. Border Top Width

You can set the width of the top border around an element with the border-top-width property.

```
border-top-width: value;
```



6. CSS Text Properties

6.1. Color

You can set the color of text with the following:

```
color: value;
```

Possible values are

- Color name - example:(red, black...)
- hexadecimal number - example:(#ff0000, #000000)
- RGB color code - example:(rgb(255, 0, 0), rgb(0, 0, 0))

6.2. Letter Spacing

You can adjust the space between letters in the following manner.

Setting the value to zero prevents the text from justifying. You can use negative values.

```
letter-spacing: value;
```

Possible values are

- normal
- length

Example:

T h e s e L e t t e r s a r e s p a c e d a t 5 p x

6.3. Text Align

You can align text with the following:

```
text-align: value;
```

Possible values are

- left
- right
- center
- justify

Examples:

This text is aligned left.

This text is aligned in the center.

This text is aligned right.

This text is justified.

6.5. Text Decoration

You can decorate text with the following:

```
text-decoration: value;
```

Possible values are

- none
- underline
- overline
- line-through

Examples:

Text Decorations set to none

This text is underlined.

This text is overlined.

~~This text has a line through it.~~

6.6. Text Indent

You can indent the first line of text in an HTML element with the following:

```
text-indent: value;
```

Possible values are

- length
- percentage

Examples:

This text is indented 30px pixels.

6.7. Text Transform

You can control the size of letters in an HTML element with the following:

```
text-transform: value;
```

Possible values are

- none
- capitalize
- lowercase
- uppercase

Examples:

This First Letter In Each Word Is Capitalized, Though It Is Not In My File.

THIS TEXT IS ALL UPPERCASE, THOUGH IT IS ALL LOWERCASE IN MY FILE.

this text is all lowercase. though it is all uppercase in my file.

6.8. White Space

You can control the whitespace in an HTML element with the following:

```
white-space: value;
```

Possible values are

- normal: Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary. This is default
- pre: Whitespace is preserved by the browser. Text will only wrap on line breaks. Acts like the `<pre>` tag in HTML
- nowrap: Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a `
` tag is encountered

6.9. Word Spacing

You can adjust the space between words in the following manner. You can use negative values.

```
word-spacing: value;
```

Values

- normal
- length

Example:

These words are spaced at 5px.

7. CSS Font Properties

7.1. Font –Family

The font-family property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems. If the browser does not support the first font, it tries the next font.

You can set what font will be displayed in an element with the font family property.

There are two choices for values:

- family-name:

The name of a **font family** of choice. In the last example, "**Gill**" and "**Helvetica**" are font families.

- generic family

The last value is a generic family name. The following **generic** families are defined:

- 'serif' (e.g., Times)
- 'sans-serif' (e.g., Helvetica)
- 'cursive' (e.g., Zapf-Chancery)
- 'fantasy' (e.g., Western)
- 'monospace' (e.g., Courier)

If you set a family name, it is best to also add the generic family at the end. As this is a prioritized list. Therefore, if the user does not have the specified font name it will use the same generic family. (See below)

```
font-family: Verdana, sans-serif;
```

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available:

7.2. Font Size

You can set the size of the text used in an element by using the font size property.

```
font-size: value;
```

There are many choices for values:

- xx-large
- x-large
- larger
- large
- medium
- small
- smaller
- x-small
- xx-small
- length
- % (percent)

7.3. Font Style

You can set the style of text in an element with the font-style property:

```
font-style: value;
```

Possible values are

- normal
- italic
- oblique

7.4. Font Variant

You can set the variant of text within an element with the font-variant property.

```
font-variant: value;
```

Possible values are

- normal
- small-caps

7.5. Font Weight

You can control the weight of text in an element with the font-weight property:

```
font-weight: value;
```

Possible values are

- lighter
- normal
- 100
- 200
- 300
- 400
- 500
- 600
- 700
- 800
- 900
- Bold
- Bolder

7.6. Font

The font property can set the style, weight, variant, size, line height and font:

```
font: italic bold normal small/1.4em Verdana, sans-serif;
```

The above would set the text of an element to an italic style a bold weight a normal variant a relative size a line height of 1.4em and the font to Verdana or another sans-serif typeface.

8. CSS Anchors, Links and Pseudo Classes

Below are the various ways you can use CSS to style links.

```
a:link {color: #009900;}  
a:visited {color: #999999;}  
a:hover {color: #333333;}  
a:focus {color: #333333;}  
a:active {color: #009900;}
```

Now let us look at what each one of the above link styles actually does.

```
a:link {color: #009900;}
```

The first on the list sets the color of a link when no event is occurring.

```
a:visited {color: #999999;}
```

The second sets the color a link changes to, when the user has already visited that URL.

```
a:hover {color: #333333;}
```

The third sets the color a link changes to as the user places their mouse pointer over the link.

```
a:focus {color: #333333;}
```

The fourth is primarily for the same purpose as the last one, but this one is for users that are not using a mouse and are tabbing through the links via their keyboards tab key, it sets the color a link changes to as the user tabs through the links.

```
a:active {color: #009900;}
```

The fifth on the list sets the color a link changes to as it is pressed.

8.1. Pseudo Classes

You can set links contained in different parts of your web page to be different colors by using the pseudo class. For example, let us say you want your links in the content area to have a different color then the links in the left or right column of your webpage.

You can do this in the following fashion:

```
#content a:link {color: #009900;}  
#content a:visited {color: #999999;}  
#content a:hover {color: #333333;}  
#content a:focus {color: #333333;}  
#content a:active {color: #009900;}
```

Now assuming that you have your main content in a division named "content" all links within that division will now be styled by this new style selector. Should your selector have a different name, just change the #content selector to match your division name.

Then for the links in a column, you could use the following:

```
#column a:link {color: #009900;}  
#column a:visited {color: #999999;}  
#column a:hover {color: #333333;}  
#column a:focus {color: #333333;}  
#column a:active {color: #009900;}
```

Once again, this assumes the name of the column division, just change the name to match yours.

This same method can be accomplished by declaring a class instead of an id.

```
a.column:link {color: #009900;}
a.column:visited {color: #999999;}
a.column:hover {color: #333333;}
a.column:focus {color: #333333;}
a.column:active {color: #009900;}
```

Though in this case you will need to add a class to each link:

```
<a class="column" href="" title="">some link text</a>
```

But, there is still yet an easier way:

```
.column a:link {color: #009900;}
.column a:visited {color: #999999;}
.column a:hover {color: #333333;}
.column a:focus {color: #333333;}
.column a:active {color: #009900;}
```

8.2. All Psudo-classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent

:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:indeterminate	input:indeterminate	Selects all undetermined <inputs>
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(language)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent

:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

9. CSS Backgrounds

9.1. Background Attachment

If you are using an image as a background. You can set whether the background scrolls with the page or is fixed when the user scrolls down the page with the background-attachment property.

```
background-attachment: value;
```

Values:

- fixed
- scroll

9.2. Background Color

You can specifically declare a color for the background of an element using the background-color property.

```
background-color: value;
```

Values:

- color name
- hexadecimal number
- RGB color code
- Transparent

9.3. Background Image

You can set an image for the background of an element using the background-image property.

```
background-image: url(path_to_image);
```

Values:

- url
- none

9.4. Background Position

You can position an image used for the background of an element using the background-position property.

```
background-position: value;
```

Values:

- top left
- top center
- top right
- center left
- center center
- center right
- bottom left
- bottom center
- bottom right
- x-% y-%
- x-pos y-pos

9.5 Background Repeat

You can set if an image set as a background of an element is to repeat (across=x and/or down=y) the screen using the background-repeat property.

```
background-repeat: value;
```

Values:

- no-repeat
- repeat
- repeat-x
- repeat-y

9.6. background Origin

Specifies the origin position (the background positioning area) of a background image.

Note: This property has no effect if **background-attachment** is **fixed**.

```
background-origin: value;
```

Value:

- padding-box: Default value. The background image starts from the upper left corner of the padding edge
- border-box: The background image starts from the upper left corner of the border
- content-box: The background image starts from the upper left corner of the content

9.7. Background Size

The background-size property specifies the size of the background images.

There are four different syntaxes you can use with this property: the keyword syntax ("**auto**", "**cover**" and "**contain**"), the one-value syntax (sets the width of the image (height becomes "auto")), the two-value syntax (first value: width of the image, second value: height), and the multiple background syntax (separated with comma).

```
background-size: value;
```

Value:

- Auto: Default value. The background image is displayed in its original size
- Length: Sets the width and height of the background image. The first value sets the width; the second value sets the height. If only one value is given, the second is set to "auto"
- Percentage: Sets the width and height of the background image in percent of the parent element. The first value sets the width; the second value sets the height. If only one value is given, the second is set to "auto"
- Cover: Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges

- Contain: Resize the background image to make sure the image is fully visible

9.8. Background

You can style the background of an element in one declaration with the background property.

```
background: #ffffff url(path_to_image) top left no-repeat fixed;
```

Values:

- attachment
- color
- image
- position
- repeat

Alternatively, you can set each property individually.

10. CSS Ordered & Unordered Lists

10.1. List Style Image

You can use an image for the bullet of unordered lists with the list-style property

```
list-style-image: url(path_to_image.gif, jpg or png);
```

If you use an image, it is a good idea to declare the list-style-type also in case the user has images turned off.

Example:

```
list-style-image: url('tikmark.gif');
```

- ✓ Item 1
- ✓ Item 2
- ✓ Item 3

10.2. List Style Position

You can control the position of ordered and unordered lists with the list style position property:

```
list-style-position: value;
```

Values

inside	outside
<ul style="list-style-type: none"> • Coffee - A brewed drink prepared from roasted coffee beans... • Tea • Coca-cola 	<ul style="list-style-type: none"> • Coffee - A brewed drink prepared from roasted coffee beans... • Tea • Coca-cola

10.3. List Style Type

You can control the type of bullet ordered and unordered lists use with the list style type property

```
list-style-type: value;
```

Values

- disc
- circle
- square
- decimal
- lower-roman
- upper-roman
- lower-alpha
- upper-alpha
- none

10.5. List Style

You can control the appearance of ordered and unordered lists in one declaration with the list-style property:

```
list-style: list-style-type list-style-position list-style-image;
```

Values:

- image
- position
- type

Alternatively, you can control them individually.

11. CSS Width and Height Properties

11.1. Height

You can control the height of an element with the height property:

```
height: value;
```

Values:

- auto
- length
- percentage

11.2. Line Height

You can control the height between lines with the line-height property

```
line-height: value;
```

Values:

- normal
- number
- length
- percentage

11.3. Max Height

You can control the maximum height of an element with the max-height property

```
max-height: value;
```

Values:

- none
- length
- percentage

11.4. Min Height

You can control the minimum height of an element with the min-height property

```
min-height: value;
```

Values:

- length
- percentage

11.5. Width

You can control the width of an element with the width property

```
width: value;
```

Values:

- auto
- length
- percentage

11.6. Max Width

You can control the maximum width of an element with the max-width property

```
max-width: value;
```

Values:

- none
- length
- percentage

11.7. Min Width

You can control the minimum width of an element with the min-width property

```
min-width: value;
```

Values:

- length
- percentage

12. CSS Positioning

12.1. Position

The position property (as you may have guessed) changes how elements are positioned on your webpage.

```
position: value;
```

Values:

- static

Static positioning is by default the way an element will appear in the normal flow of your HTML file. It is not necessary to declare a position of static. Doing so, is no different than not declaring it at all.

```
position: static;
```

- Relative

Positioning an element relatively places the element in the normal flow of your HTML file and then offsets it by some amount using the properties left, right, top and bottom. This may cause the element to overlap other elements that are on the page, which of course may be the effect that is required.

```
position: relative;
```

- Absolute

Positioning an element absolutely, removes the element from the normal flow of your HTML file, and positions it to the top left of its nearest parent element that has a position declared other than static. If no parent element with a position other than static exists then it will be positioned from the top left of the browser window.

```
position: absolute;
```


- fixed

Positioning an element with the fixed value is the same as absolute except the parent element is always the browser window. It makes no difference if the fixed element is nested inside other positioned elements.

Furthermore, an element that is positioned with a fixed value will not scroll with the document. It will remain in its position regardless of the scroll position of the page.

```
position: fixed;
```

When positioning elements with relative, absolute or fixed values the following properties are used to offset the element:

- top
- left
- right
- bottom

```
position: absolute; top: 10px; right: 10px;
```

13. CSS Classification

13.1. Float

The float property changes how text and or images within an element are displayed

```
float: value;
```

Values:

- Left

The image/text is displayed to the left of the parent element

- Right

The image/text is displayed to the right of the parent element

- None

There is no change in the way the image/text is displayed

13.2. Display

You can control how an element is displayed with the display property

```
display: value;
```

- block

Creates a line break before and after the element

- inline

No line break is created

- inline-block

Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values

- list-item

Creates a line break before and after the element and adds a list item marker

- none

Makes an element not display on the page and empty its space

13.3. Clear

The clear property specifies on which sides of an element floating elements are not allowed to float.

```
clear: value;
```

Values:

- none

This is the default setting; floated elements can appear on either side of the element set to clear: none;

- both

Setting the value to both, causes no floated elements to appear on either side of the element set to clear: both;

- left

Setting the value to left, causes no floated elements to appear to the left side of the element set to clear: left;

- right

Setting the value to right, causes no floated elements to appear to the right side of the element set to clear: right;

13.4. Clip

You can control how much of an element is visible with the clip property

```
position: absolute;  
clip: value;
```

Values:

- auto
- shape

Currently the only shape recognized by the clip property is **rect** (rectangle)

























```
clip: rect(<top>, <right>, <bottom>, <left>);
```

13.5. Cursor

You can control the style of cursor to be used in an element with the cursor property

```
cursor: value;
```

Values:

 auto	 move	 no-drop	 col-resize
 all-scroll	 pointer	 not-allowed	 row-resize
 crosshair	 progress	 e-resize	 ne-resize
 default	 text	 n-resize	 nw-resize
 help	 vertical-text	 s-resize	 se-resize
 inherit	 wait	 w-resize	 sw-resize

If you choose to use a custom cursor, it is always a good idea to declare a generic one after the custom value.

```
cursor: url("image.cur"), default;
```

13.6. Overflow

You can control what an elements contents will do if it overflows it boundaries with the overflow property

```
overflow: value;
```

Values:

- auto
Put scroll bar as it is required
- hidden
Hide extra text or content
- visible

Default value, overflow without handling

- scroll

Scroll both X and Y

13.7. Visibility

You can control if an element is visible or not with the visibility property

```
visibility: value;
```

Values:

- hidden

Hide element but remains its space

- visible

Default value

13.8. Z-Index

You can control the layer order of positioned elements with the z-index property

```
z-index: value;
```

Values:

- auto
- number

The higher the number the higher the level. Negative numbers are allowed

14. Visibility Problem

There are three ways to change the visibility of an object and make it hidden:

14.1. Visibility property

You can control if an element is visible or not with the visibility property

```
visibility: value;
```

Values:

- hidden
Hide element but remains its space
- visible
Default value

14.2. Display property

You can control how an element is displayed with the display property

```
display: value;
```

- none
Makes an element not display on the page and empty its space

14.3. Opacity Property

The opacity property sets the opacity level for an element.

The opacity-level describes the transparency-level, where 1 is not transparent at all, 0.5 is 50% see-through, and 0 is completely transparent.

```
opacity: value;
```

Values: number (a number between 0: invisible to 1: fully visible)



Opacity: 0.2



Opacity: 0.5



Opacity: 1

15. CSS Pseudo Elements

15.1. The Syntax

The syntax for pseudo elements is a bit different than that of regular CSS, but it is real close.

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

As you can see the only difference is that you place the pseudo element after the selector, and divide the 2 with a (:) colon.

The elements:

- first-line
- first-letter

15.2. First Line

The first-line pseudo element styles the first line of text in a block level element.

```
p{font-size: small;}
p:first-line {font-size: medium; color: #ff0000;}
```

As you can see in the above example paragraphs are set to be a small font size, but the `p:first-line` is set to be a medium size and a red color.

The result is that the first line of all paragraphs will be red in color and a bit larger than the rest of the paragraph.

Though let us say you only want to style a certain paragraph of text with the `first-line` element. That is where declaring a class to the pseudo element comes into play.

15.3. First-line with class

```
p.special:first-line {font-size: medium; color: #ff0000;}
```

I have declared a class of special within my css file.

The following properties can be assigned to the first-line pseudo

- element:
- background
- clear
- color
- font
- letter-spacing
- line-height
- text-decoration
- text-transform
- vertical-align
- word-spacing

15.4. First Letter

The first-letter pseudo element styles the first letter of text in a block level element.

```
p{font-size: small;}
p:first-letter {font-size: medium; color: #ff0000;}
```

As you can see in the above example paragraphs are set to be a small font size, but the `p:first-letter` is set to be a medium size and a red color. The result is that the first letter of all paragraphs will be red in color and a bit larger than the rest of the paragraph.

Though let us say you only want to style a certain paragraph of text with the `first-letter` element. That is where declaring a class to the pseudo element comes into play.

15.5. First-letter with class

```
p.special_letter:first-letter {font-size: x-large; font-weight:
                                bold; color: #ff0000;}
```

I have declared a class of special_letter within my css file.

The following properties can be assigned to the first-letter pseudo element:

- background

- border
- clear
- color
- float
- font
- line-height
- margin
- padding
- text-decoration
- text-transform
- word-spacing

15.6. ::before

The `::before` pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each `<h1>` element:

```
h1::before {  
  content: url(smiley.gif);  
}
```

15.7. ::after

The `::after` pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after the content of each `<h1>` element:

```
h1::after {  
  content: url(smiley.gif);  
}
```

Hint: The content property is used with the `::before` and `::after` pseudo-elements, to insert generated content.

15.8. ::selection

The `::selection` pseudo-element matches the portion of an element that is selected by a user.

The following example makes the selected text red on a yellow background:

```
::selection {  
  color: red;  
  background: yellow;  
}
```

15.9. ::placeholder

The `::placeholder` selector selects form elements with placeholder text.

The placeholder text is set with the placeholder attribute, which specifies a hint that describes the expected value of an input field.

Hint: The default color of the placeholder text is light grey in most browsers.

```
::-webkit-input-placeholder { /* Edge */  
  color: red;  
}  
  
:-ms-input-placeholder { /* Internet Explorer 10-11 */  
  color: red;  
}  
  
::placeholder {  
  color: red;  
}
```

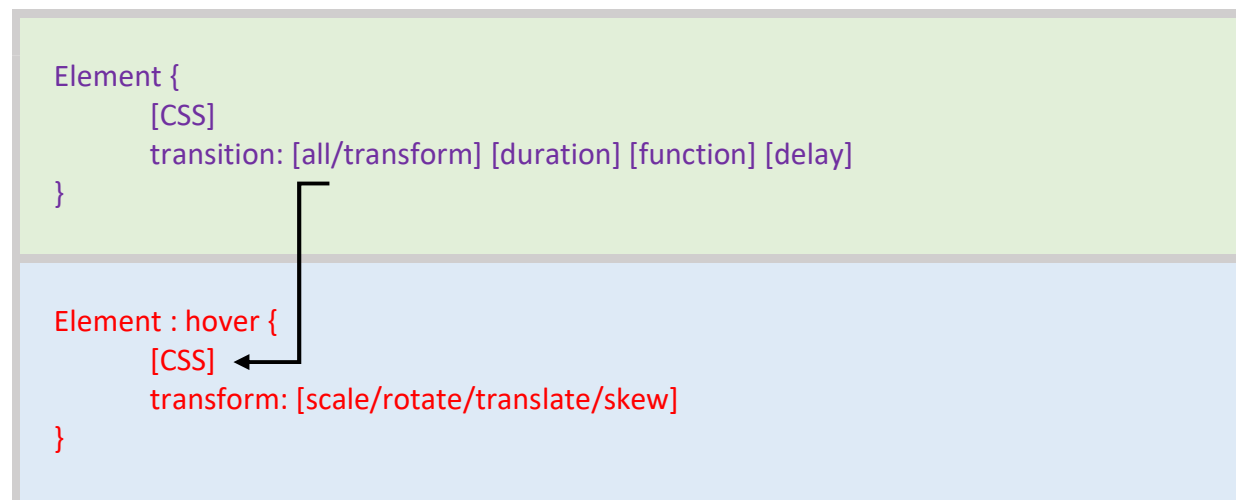
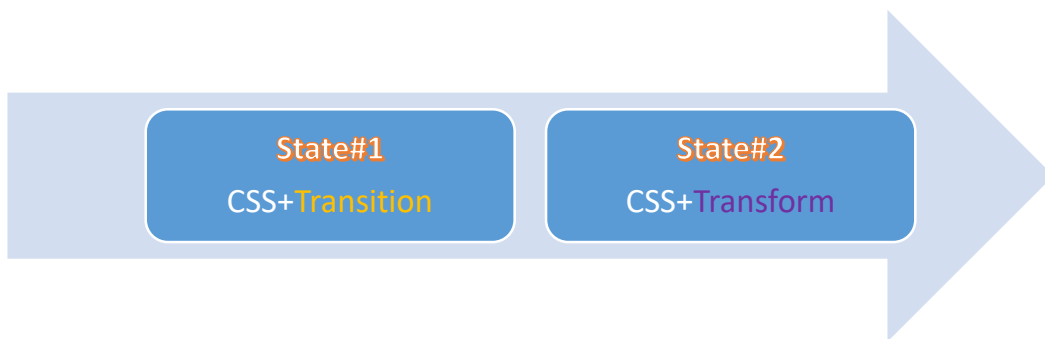
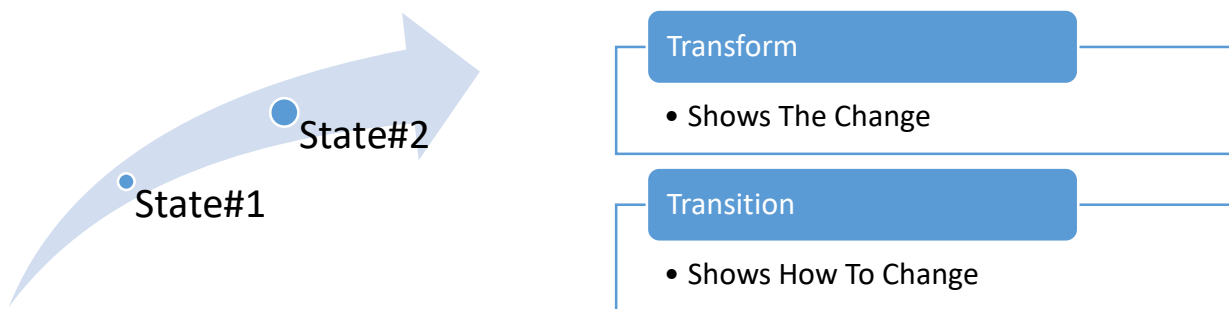
16. CSS Transition and Transform

Here we will introduce you to CSS transitions and CSS transforms: **the CSS power couple**. When used together, these properties allow you to create simple animations and add valuable interaction and visual feedback for your users.

So what are transforms and transitions? At their most basic level,

Transforms move or change the appearance of an element, while

Transitions make the element smoothly and gradually change from one state to another.



16.1. CSS transitions: an introduction

Let us start with CSS transitions. Transitions are the grease in the wheel of CSS transforms. Without a transition, an element being transformed would change abruptly from one state to another. By applying a transition, you can control the change, making it smooth and gradual.

Two properties are required in order for the transition to take effect:

```
transition-property
transition-duration
```

Hint: Transition Shorthand

Each transition property can be defined individually, but for cleaner and faster code, it is recommended that you use the transition shorthand.

Here is the full shorthand sequence. Again, the first two properties are required.

```
div {
  transition:    [property]    [duration]    [timing-function]
               [delay];
}
```

16.1.1. transition-property (required):

The `transition-property` specifies the CSS property where the transition will be applied. You may apply a transition to an individual property (e.g., `background-color` or `transform`) or to all properties in the rule-set (i.e., `all`).

```
div {
  transition-property: all;
--or--
  transition-property: transform;
}
```

16.1.2. transition-duration (required):

The `transition-duration` property specifies the time span of the transition. You can specify in seconds or milliseconds.

```
div {  
  transition-duration: 3s;  
}
```

Hint: Shorthand example for transition-duration

```
div {  
  transition: all 3s;  
}
```

16.1.3. transition-timing (optional):

The `transition-timing-function` property allows you to define the speed of the transition over the duration.

The default timing is `ease`, which starts out slow, quickly speeds up, and then slows down at the end. The other timing options are:

- linear
- ease
- ease-in
- ease-out
- ease-in-out

16.1.4. transition-delay (optional):

The `transition-delay` property allows you to specify when the transform will start.

By default, the transition starts as soon as it is triggered (e.g., on mouse hover). However, if you want to transition to start after it is triggered you can use the transition delay property.

Hint: Shorthand example for transition-delay

```
div {  
  transition: all 3s 1s;  
}
```

16.2. CSS transforms: an introduction

Now that we reviewed how to make smooth and gradual transitions, let us look at CSS transforms - how to make an element change from one state to another. With the CSS transform property you can

- rotate
- move
- skew
- scale

Elements.

Transforms are triggered when an element changes states, such as on mouse-hover or mouse-click.

16.2.1. scale

Allows you to increase or decrease the size of an element.

For example, the value 2 would transform the size to be 2 times its original size. The value 0.5 would transform the size to be half its original size.

CSS:

```
.square {
    background: darkturquoise;
    border-radius: 5px;
    height: 100px;
    margin: 100px;
    transition: transform 1s;
    width: 100px;
}

.square:hover{
    transform: scale(2);
}
```

HTML:

```
<div class="square"></div>
```

You can scale an element by setting parameters for the width (X-axis) or height (Y-axis):

```
.square:hover{
    transform: scaleX(2);
    transform: scaleY(0.5);
}
```

Or, use the `scale()` shorthand to scale both axes at the same time:

```
.square:hover{
    transform: scale(2, 0.5);
}
```

16.2.2. rotate

With the `rotate` value, the element rotates clockwise or counterclockwise by a specified number of degrees. A positive value, such as `90deg`, rotates the element **clockwise**, while a negative value, such as `-90deg`, rotates it **counterclockwise**.

CSS:

```
.droplet {
    border-radius: 2% 50%;
    height: 100px;
    margin: 100px;
    transition: all 3s;
    transition-timing: ease-in-out;
    width: 100px;
}

.droplet:hover {
    transform: rotate(1080deg);
}
```

HTML:

```
<div class="droplet"></div>
```

You can rotate more than a full rotation with numbers over than 360, such as `1080deg`, for three full rotations.

16.2.3. translate

The `translate` value moves an element left/right and up/down. The movement is based on the parameters given for the **X (horizontal)**-**Y (vertical)** axes.

A positive X value moves the element to the right, while a negative X moves the element to the left. A positive Y value moves the element downwards and a negative Y value, upwards.

CSS:

```
.square {
  background: #2b3f53;
  border-radius: 3px;
  height: 150px;
  margin: 100px;
  transition: transform 0.8s;
  position: absolute;
  width: 150px;
}
.square:hover {
  transform: translate(20px, 20px);
}
```

HTML:

```
<div class="square"></div>
```

16.2.4. skew

With the skew value, the element skews (or tilts) one direction or the other based on the values given for the **X** and **Y axes**.

CSS:

```
.square {
  background: Khaki;
  border-radius: 5px;
  height: 150px;
  margin: 100px;
  transition: transform 1s;
  width: 150px;
}
.square:hover {
  transform: skewX(-20deg);
}
```

HTML:

```
<div class="square"></div>
```

A positive X value tilts the element left, while a negative X value tilts it right. A positive Y value tilts the element down, and a negative Y value tilts is up.

Or use a shorthand to include both X and Y properties:


```
div {
  transform: skewX(25deg);
  transform: skewY(10deg);
  transform: skew(25deg, 10deg);
}
```

Hint: Skewing an element will also skew all of the children inside of the element as well. If you need to maintain the original angle of a child element, you can use the opposite value of skew to bring it back.

16.2.5. transform-origin

The transform-origin property is separate from the transform property but works in tandem with it. It allows you to specify the location origin of the transform. **By default, the origin is in the center of the element.**

For example, if you are using the transform: rotate property but want it to rotate not from the center, but from the **top left** corner, you'd use the value **0% 0%** or **left top**. For the **bottom right** corner, you would use **100% 100%** or **right bottom**, etc.

CSS:

```
.droplet {
  background: skyblue;
  border-radius: 5px 50%;
  height: 100px;
  margin: 150px auto;
  transform-origin: left top;
  transition: transform 3s;
  width: 100px;
}

.droplet:hover {
  transform: rotate(500deg);
}
```

HTML:

```
<div class="droplet"></div>
```

Make sure to add the transform-origin property to the parent element, not with the transform property in the hover selector.

16.3. Combining transforms

The transform shorthand allows you to string the various transform methods into one property.

```
div {  
  transform: rotate(90deg) scale(2) translateY(-50%)  
  translateX(50%);  
}
```

17. CSS Animation

17.1. The Building Blocks of Animations

CSS animations are made up of two basic building blocks:

1. **Keyframes** - define the stages and styles of the animation.
2. **Animation Properties** - assign the `@keyframes` to a specific CSS element and define *how* it is animated.

Let us look at each individually.

<pre> @keyframes [Name] { 0% { [CSS] transform: [scale/rotate/translate/skew] } 60% { [CSS] transform: [scale/rotate/translate/skew] } 100% { [CSS] transform: [scale/rotate/translate/skew] } } </pre>	<pre> Element{ Animation: [Name] [Duration] [Function] [Delay] } </pre>
---	---

17.1.1. Keyframes

Keyframes are the foundation of CSS animations. They define what the animation looks like at each stage of the animation timeline. Each `@keyframes` is composed of:

- **Name of the animation:** A name that describes the animation, for example, `bounceIn`.
- **Stages of the animation:** Each stage of the animation is represented as a percentage. `0%` represents the beginning state of the animation. `100%` represents the ending state of the animation. Multiple intermediate states can be added in between.
- **CSS Properties:** The CSS properties defined for each stage of the animation timeline.

Let us take a look at a simple `@keyframes` I've named `bounceIn`. This `@keyframes` has three stages:

1. At the first stage (0%), the element is at opacity 0 and scaled down to 10 percent of its default size, using CSS transform scale
2. At the second stage (60%) the element fades in to full opacity and grows to 120 percent of its default size
3. At the final stage (100%), it scales down slightly and returns to its default size

The `@keyframes` are added to your main CSS file:

```
@keyframes bounceIn {
  0% {
    transform: scale(0.1);
    opacity: 0;
  }
  60% {
    transform: scale(1.2);
    opacity: 1;
  }
  100% {
    transform: scale(1);
  }
}
```

17.1.2. Animation Properties

Once the `@keyframes` are defined, the animation properties must be added in order for your animation to function.

Animation properties do two things:

1. They assign the `@keyframes` to the elements that you want to animate.
1. They define *how* it is animated.

The animation properties are added to the CSS selectors (or elements) that you want to animate. You must add the following two animation properties for the animation to take effect:

- `animation-name`: The name of the animation, defined in the `@keyframes`.
- `animation-duration`: The duration of the animation, in seconds (e.g., 5s) or milliseconds (e.g., 200ms).

Continuing with the above `bounceIn` example, we will add `animation-name` and `animation-duration` to the div that we want to animate.

```
div {  
  animation-duration: 2s;  
  animation-name: bounceIn;  
}
```

Shorthand syntax:

```
div {  
  animation: bounceIn 2s;  
}
```

17.2. Additional Animation Properties

In addition to the required `animation-name` and `animation-duration` properties, you can further customize and create complex animations using the following properties:

- `animation-timing-function`
- `animation-delay`
- `animation-iteration-count`
- `animation-direction`
- `animation-fill-mode`
- `animation-play-state`

17.2.1. Animation-timing-function

Defines the speed curve or pace of the animation.

```
animation-timing-function: ease-in-out;
```

You can specify the timing with the following predefined timing options:

- `ease`
- `linear`
- `ease-in`
- `ease-out`
- `ease-in-out`

The default value, if no other value is assigned, is `ease`, which starts out slow, speeds up, then slows down.

Hint: Animation shorthand syntax (recommended):

```
animation: [animation-name] [animation-duration] [animation-timing-function];
animation: bounceIn 2s ease-in-out;
```

17.2.2. Animation-Delay

Allows you to specify when the animation (or pieces of the animation) will start. A positive value (such as 2s) will start the animation 2 seconds after it is triggered. The element will remain unanimated until that time. A negative value (such as -2s) will start the animation at once, but starts 2 seconds into the animation.

```
animation-delay: 5s;
```

The value is defined in seconds (s) or milliseconds (mil).

Hint: Animation shorthand syntax (recommended):

```
animation: [animation-name] [animation-duration] [animation-timing-function]
[animation-delay];
animation: bounceIn 2s ease-in-out 3s;
```

17.2.3. Animation-iteration-count

Specifies the number of times that the animation will play. The possible values are:

- Number - a specific number of iterations (default is 1)
- Infinite - the animation repeats forever

```
animation-iteration-count: 2;
```

Hint: Animation shorthand syntax (recommended):

```
animation: [animation-name] [animation-duration] [animation-timing-function]
[animation-delay] [animation-iteration-count];
animation: bounceIn 2s ease-in-out 3s 2;
```

17.2.4. Animation-direction

Specifies whether the animation should play forward, reverse, or in alternate cycles.

```
animation-direction: alternate;
```

The possible values are:

- **normal** (default) - The animation plays forward. On each cycle the animation resets to the beginning state (0%) and plays forward again (to 100%).
- **reverse** - The animation plays backwards. On each cycle the animation resets to the end state (100%) and plays backwards (to 0%).
- **alternate** - The animation reverses direction every cycle. On each odd cycle, the animation plays forward (0% to 100%). On each even cycle, the animation plays backwards (100% to 0%).
- **alternate-reverse** - The animation reverses direction every cycle. On each odd cycle, the animation plays in reverse (100% to 0%). On each even cycle, the animation plays forward (0% or 100%).

Hint: Animation shorthand syntax (recommended):

```
animation: [animation-name] [animation-duration] [animation-timing-function]
[animation-delay] [animation-iteration-count] [animation-direction];
animation: bounceIn 2s ease-in-out 3s 3 alternate;
```

17.2.5. Animation-fill-mode

Specifies if the animation styles are visible before or after the animation plays. This property is a little confusing, but once understood it is very useful.

By default, the animation will not affect the styles of the element before the animation begins (if there is an animation-delay) or after the animation is finished. The animation-fill-mode property can override this behavior with the following possible values:

- **backwards** - Before the animation (during the animation delay), the styles of the initial keyframe (0%) are applied to the element.
- **forwards** - After the animation is finished, the styles defined in the final keyframe (100%) are retained by the element.
- **both** - The animation will follow the rules for both forwards and backwards, extending the animation properties before and after the animation.

- normal (default) - The animation does not apply any styles to the element, before or after the animation.

```
animation-fill-mode: forwards;
```

Hint: Animation shorthand syntax (recommended):

```
animation: [animation-name] [animation-duration] [animation-timing-  
function] [animation-delay] [animation-iteration-count] [animation-  
direction] [animation-fill-mode];  
animation: bounceIn 2s ease-in-out 3s 3 forwards;
```

17.2.6. Animation-play-state

The `animation-play-state` specifies whether the animation is playing or paused. Resuming a paused animation starts the animation where it was left off.

The possible values are:

- playing - The animation is currently running
- paused - The animation is currently paused

Example:

```
.div:hover {  
  animation-play-state: paused;  
}
```

17.2.7. Multiple Animations

To add multiple animations to a selector, you simply separate the values with a comma. Here's an example:

```
.div {  
  animation: slideIn 2s, rotate 1.75s;  
}
```

17.3. Note About Prefixes

As of late 2014, many Webkit based browsers still use the `-webkit-`prefixed version of both animations, keyframes, and transitions. Until they adopt the standard version, you'll want to include both unprefixed and Webkit versions in your code.

Keyframes and animations with WebKit prefixes:

```
div {  
  -webkit-animation-duration: 2s;  
  animation-duration: 2s;  
  -webkit-animation-name: bounceIn;  
  animation-name: bounceIn;  
}
```

```
@-webkit-keyframes bounceIn { /* styles */ }  
@keyframes bounceIn { /* styles */ }
```