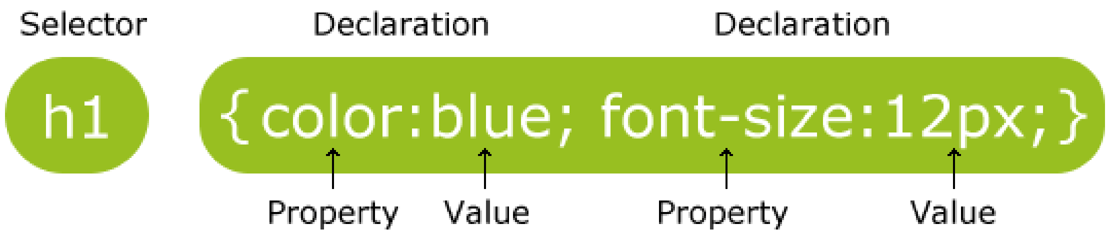
2012-03-02　　Edited by: xuzhezhao　　Last edited: 2013-03-05

CSS 学习笔记

1. CSS Syntax

CSS rule has two main parts: a **selector**, and one or more **declarations**:



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

A CSS **comment** begins with "**/***", and ends with "***/**".

2. CSS Id and Class

- **id Selector**
  The **id** selector is used to specify a style for **a single, unique element.**

  The id selector uses the id attribute of the HTML element, and is defined with a **"#"**.

  The style rule below will be applied to the element with id="para1":

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

- **The class Selector**

  The **class** selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements.

  This allows you to set a particular style for many HTML elements with the same class.

  The class selector uses the HTML class attribute, and is defined with a "."

  In the example below, all HTML elements with class="center" will be center-aligned:

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

  You can also specify that o**nly specific HTML elements** should be affected by a class.

  In the example below, all p elements with class="center" will be center-aligned:

  ```
  p.center {text-align:center;}
  ```

3. Three Ways to Insert CSS

- **External Style Sheet**

  An external style sheet is ideal when the style is applied to **many pages.** With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the **<link>** tag. The **<link>** tag goes inside the head section:

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

NOTE: Do not add a space between the property value and the unit (such as margin-left:20 px). The correct way is: margin-left: 20px

- **Internal Style Sheet**

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, by using the **<style>** tag, like this:

```
<head>
<style>
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

- **Inline Styles**

  An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly!

  ```
  <p style="color:sienna;margin-left:20px">This is a paragraph.</p>
  ```

4. Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.
For example, an external style sheet has these properties for the h3 selector:

```
h3
{
color:red;
text-align:left;
font-size:8pt;
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
text-align:right;
font-size:20pt;
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color:red;
text-align:right;
font-size:20pt;
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

Multiple Styles Will Cascade into One.

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. **Browser default**
2. **External style sheet**
3. **Internal style sheet (in the head section)**
4. **Inline style (inside an HTML element)**

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

5. CSS Background

- **Background Color**

  The background color of a page is defined in the body selector:

  ```
  body {background-color:#b0c4de;}
  ```

  With CSS, a color is most often specified by:

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

- **Background Image**

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

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

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

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

  - **Repeat Horizontally or Vertically**

    ```
    body
    {
    background-image:url('gradient2.png');
    ```

```
background-repeat:repeat-x;
}
```

- **Set position and no-repeat**

```
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
background-position:right top;
}
```

- **Shorthand property**

```
body {background:#ffffff url('img_tree.png') no-repeat right top;}
```

When using the shorthand property the order of the property values is:

- **background-color**
- **background-image**
- **background-repeat**
- **background-attachment**
- **background-position**

It does not matter if one of the property values is missing, as long as the ones that are present are in this order.

6. CSS Text

- **Text Color**

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

- **Text Alignment**

```
h1 {text-align:center;}
p.date {text-align:right;}
p.main {text-align:justify;}
```

- **Text Decoration**

  The text-decoration property is mostly used to remove **underlines** from links for design purposes:

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

h1 {text-decoration:overline;}
h2 {text-decoration:line-through;}
h3 {text-decoration:underline;}
h4 {text-decoration:blink;}
```

- **Text Transformation**

  It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

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

- **Text Indentation**

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

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

7. CSS Font

- **CSS Font Families**

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

- **Font Style**

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

- **Font Size**

  ```
  h1 {font-size:40px;}
  h2 {font-size:30px;}
  p {font-size:14px;}
  ```

  The size can be calculated from pixels to em using this formula: **pixels/16=em**

  ```
  h1 {font-size:2.5em;} /* 40px/16=2.5em */
  h2 {font-size:1.875em;} /* 30px/16=1.875em */
  p {font-size:0.875em;} /* 14px/16=0.875em */
  ```

  ◆ **Use a Combination of Percent and Em**

  ```
  body {font-size:100%;}
  ```

```
h1 {font-size:2.5em;}
h2 {font-size:1.875em;}
p {font-size:0.875em;}
```

8. CSS Links

- **Common Link Styles**
  - **Text Decoration**

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

  - **Background Color**

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

9. CSS Lists

- **Different List Item Markers**

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

ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

- **An Image as The List Item Marker**

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

10. CSS Tables

- **Table Borders**

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

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

- **Collapse Borders**

  The border-collapse property sets whether the table borders are collapsed into a single border or separated:

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

- **Table Width and Height**

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

- **Table Text Alignment**

  The text in a table is aligned with the **text-align** and **vertical-align** properties.

  The **text-align** property sets the horizontal alignment, like **left, right, or center:**

```
td
{
text-align:right;
}
```

  The **vertical-align** property sets the vertical alignment, like top, bottom, or middle:

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

- **Table Padding**
  To control **the space between the border and content** in a table, use the **padding** property on td and th elements:

```
td
{
padding:15px;
}
```

- **Table Color**
  The example below specifies the color of the borders, and the text and background color of th elements:

```
table, td, th
{
border:1px solid green;
}
th
{
background-color:green;
color:white;
}
```

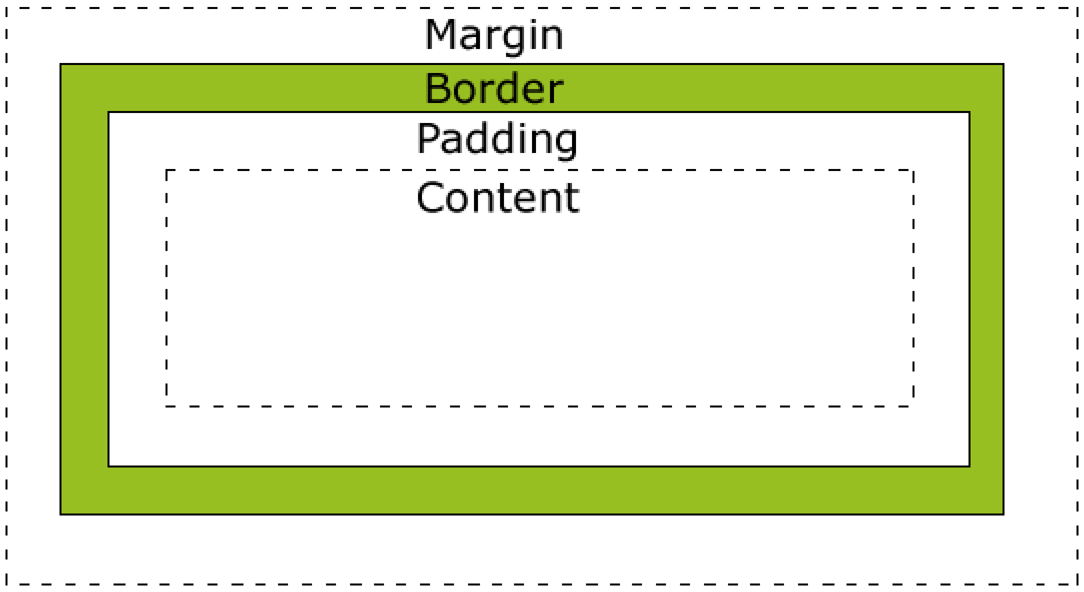
11. CSS Box Model

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

The CSS box model is essentially a box that wraps around HTML elements, and it consists of: **margins**, **borders**, **padding**, and the **actual content.**

The box model allows us to place a border around elements and space elements in relation to other elements.

The image below illustrates the box model:



Explanation of the different parts:

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent
- **Border** - A border that goes around the padding and content. The border is affected by the background color of the box
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box
- **Content** - The content of the box, where text and images appear

- **Width and Height of an Element**

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

  The total width of the element in the example below is 300px:

  width:250px;
  padding:10px;
  border:5px solid gray;
  margin:10px;

12. CSS Border

- **Border Style**

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

Note: None of the border properties will have ANY effect unless the **border-style** property is set!

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

- **Border Width**

  The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

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

- **Border Color**

  The **border-color** property is used to set the color of the border. The color can be set by:

- **name** - specify a color name, like "red"
- **RGB** - specify a RGB value, like "rgb(255,0,0)"
- **Hex** - specify a hex value, like "#ff0000"

You can also set the border color to "**transparent**".

```
p.one
{
border-style:solid;
border-color:red;
}
p.two
{
border-style:solid;
border-color:#98bf21;
}
```

- **Border - Individual sides**

  In CSS it is possible to specify different borders for different sides:

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

- **Border - Shorthand property**
  The border property is a shorthand for the following individual border properties:

  - border-width
  - border-style (required)
  - border-color

```
border:5px solid red;
```

13. CSS Outlines

An outline is a line that is drawn **around elements** (**outside the borders**) to make the element "stand out".

Property	Description	Values	CSS
outline	Sets all the outline properties in one declaration	*outline-color* *outline-style* *outline-width* inherit	2
outline-color	Sets the color of an outline	*color_name* *hex_number* *rgb_number* invert inherit	2
outline-style	Sets the style of an outline	none dotted dashed solid double groove ridge inset outset inherit	2
outline-width	Sets the width of an outline	thin medium thick *length* inherit	2

14. CSS Margin

The **margin clears an area around an element** (outside the border). The margin does not have a background color, and is completely **transparent**.

Value	**Description**
auto	The browser calculates a margin
length	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent of the width of the containing element
inherit	Specifies that the margin should be inherited from the parent element

- **Margin - Individual sides**

margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;

- **Margin - Shorthand property**

The margin property can have from one to four values.

- **margin:25px 50px 75px 100px;**
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px
- **margin:25px 50px 75px;**
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px
- **margin:25px 50px;**
  - top and bottom margins are 25px
  - right and left margins are 50px
- **margin:25px;**
  - all four margins are 25px

15. CSS Padding

The padding clears **an area around the content** (**inside the border**) of an element. The padding is affected by the background color of the element.

- **Padding - Individual sides**

padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;

- **Padding - Shorthand property**

The same as margin!

16. CSS Grouping and Nesting Selectors

- **Grouping Selectors**
  ```
  h1,h2,p
  {
  color:green;
  }
  ```

- **Nesting Selectors**
  It is possible to apply a style for a selector within a selector.

  In the example below, **one style is specified for all p elements, one style is specified for all elements with class="marked",** and **a third style is specified only for p elements within elements with class="marked":**
  ```
  p
  {
  color:blue;
  text-align:center;
  }
  .marked
  {
  background-color:red;
  }
  .marked p
  {
  color:white;
  }
  ```

17. CSS Dimension

The number in the "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Property	Description	Values	CSS
height	Sets the height of an element	auto *length* *%* inherit	1
max-height	Sets the maximum height of an element	none	2

		length % inherit	
max-width	Sets the maximum width of an element	none *length* % inherit	2
min-height	Sets the minimum height of an element	*length* % inherit	2
min-width	Sets the minimum width of an element	*length* % inherit	2
width	Sets the width of an element	auto *length* % inherit	1

18. CSS Display and Visibility

- **Hiding an Element - display:none or visibility:hidden**

Hiding an element can be done by setting the **display** property to "**none**" or the **visibility** property to "**hidden**". However, notice that these two methods produce **different results:**

visibility:hidden hides an element, but it will still **take up the same space** as before. The element will be hidden, but still **affect the layout.**
display:none hides an element, and it will **not take up any space**. The element will be hidden, and the page will be displayed as if the element is not there.

- **CSS Display - Block and Inline Elements**

A block element is an element that takes up the full width available, and **has a line break before and after it.**

Examples of **block elements**:

- **`<h1>`**
- **`<p>`**

- `<div>`

An **inline element** only takes up as much width as necessary, and does not force line breaks.

Examples of **inline elements:**

- `<span>`
- `<a>`

- **Changing How an Element is Displayed**

  Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow web standards.

  The following example displays list items as inline elements:
  li {display:inline;}

  The following example displays span elements as block elements:
  span {display:block;}

19. CSS Positioning

Decide which element to display in front! **Elements can overlap!**

- **Positioning**

  Elements can be positioned using the **top, bottom, left, and right** properties. However, these properties will not work unless the position property is set first. They also work differently depending on the positioning method.

- **Static Positioning**

  HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.

  Static positioned elements are not affected by the top, bottom, left, and right properties.

- **Fixed Positioning**

  An element with fixed position is positioned **relative to the browser window.**

  It will not move even if the window is scrolled:
  p.pos_fixed

```
{
position:fixed;
top:30px;
right:5px;
}
```

Fixed positioned elements can overlap other elements.

- **Relative Positioning**

  A relative positioned element is positioned **relative to its normal position.**

```
h2.pos_left
{
position:relative;
left:-20px;
}
h2.pos_right
{
position:relative;
left:20px;
}
```

The content of relatively positioned elements **can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.**

- **Absolute Positioning**

  An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

```
h2
{
position:absolute;
left:100px;
top:150px;
}
```

Absolutely positioned elements are removed from the normal flow. The document and **other elements behave like the absolutely positioned element does not exist. Absolutely positioned elements can overlap other elements.**

- **Overlapping Elements**

  When elements are positioned outside the normal flow, they can overlap other elements.

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

An element can have a positive or negative stack order:

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

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap, without a z-index specified, the element positioned last in the HTML code will be shown on top.

20. CSS Float

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.

Float is very often used for images, but it is also useful when working with layouts.

Elements are floated horizontally, this means that an element can only be floated left or right, not up or down.

A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.

The elements after the floating element will flow around it.

The elements before the floating element will not be affected.

If an image is floated to the right, a following text flows around it, to the left:

```
img
{
float:right;
}
```

- **Turning off Float - Using Clear**

```
text_line
{
clear:both;
```

}

Property	Description	Values
clear	Specifies which sides of an element where other floating elements are not allowed	left right both none inherit
float	Specifies whether or not a box should float	left right none inherit

21. CSS Horizontal Align

- **Center Aligning Using the margin Property**
  Block elements can be aligned by setting the left and right margins to "**auto**".

```
.center
{
margin-left:auto;
margin-right:auto;
width:70%;
background-color:#b0e0e6;
}
```

- **Left and Right Aligning Using the position Property**
  One method of aligning elements is to use **absolute positioning:**

```
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

- **Left and Right Aligning Using the float Property**

```
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

22. CSS Pseudo-classes

- **Syntax**

  The syntax of pseudo-classes:

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

  CSS classes can also be used with pseudo-classes:

  ```
  selector.class:pseudo-class {property:value;}
  ```

- **Anchor Pseudo-classes**

  ```
  a:link {color:#FF0000;}           /* unvisited link */
  a:visited {color:#00FF00;}    /* visited link */
  a:hover {color:#FF00FF;}    /* mouse over link */
  a:active {color:#0000FF;}    /* selected link */
  ```

- **Pseudo-classes and CSS Classes**

  Pseudo-classes can be combined with CSS classes:

  ```
  a.red:visited {color:#FF0000;}
  <a class="red" href="css_syntax.asp">CSS Syntax</a>
  ```

- **CSS - The :first-child Pseudo-class**

  ◆ **Match the first <p> element**

  In the following example, the selector matches any <p> element that is the **first child** of any element:

  ```
  <html>
  <head>
  <style>
  p:first-child
  {
  color:blue;
  }
  </style>
  </head>

  <body>
  <p>I am a strong man.</p>
  <p>I am a strong man.</p>
  </body>
  </html>
  ```

- **Match the first <i> element in all <p> elements**

  In the following example, the selector matches the first <i> element in all <p> elements:

```
<html>
<head>
<style>
p > i:first-child
{
color:blue;
}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

- **Match all <i> elements in all first child <p> elements**

  In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

```
<html>
<head>
<style>
p:first-child i
{
color:blue;
}
</style>
</head>

<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

- **CSS - The :lang Pseudo-class**

  The **:lang pseudo-class** allows you to define special rules for different languages.

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

```
q:lang(no) {quotes: "~" "~";}
</style>
</head>

<body>
<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>
</body>
</html>
```

Selector	Example	Example description
:link	a:link	Selects all unvisited links
:visited	a:visited	Selects all visited links
:active	a:active	Selects the active link
:hover	a:hover	Selects links on mouse over
:focus	input:focus	Selects the input element which has focus
:first-letter	p:first-letter	Selects the first letter of every <p> element
:first-line	p:first-line	Selects the first line of every <p> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:before	p:before	Insert content before every <p> element
:after	p:after	Insert content after every <p> element
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"

23. CSS Pseudo-elements

- **Syntax**
  The syntax of pseudo-elements:
  `selector:pseudo-element {property:value;}`

  CSS classes can also be used with pseudo-elements:
  `selector.class:pseudo-element {property:value;}`

- **The :first-line Pseudo-element**
  The "**first-line**" pseudo-element is used to add a special style to the first

line of a text.

In the following example the browser formats the first line of text in a **p** element according to the style in the "**first-line**" pseudo-element (**where the browser breaks the line, depends on the size of the browser window**):

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

Note: The "first-line" pseudo-element can only be used with block-level elements.

Note: The following properties apply to the "first-line" pseudo-element:

- **font properties**
- **color properties**
- **background properties**
- **word-spacing**
- **letter-spacing**
- **text-decoration**
- **vertical-align**
- **text-transform**
- **line-height**
- **clear**

- **The :first-letter Pseudo-element**

```
p:first-letter
{
color:#ff0000;
font-size:xx-large;
}
```

- **Pseudo-elements and CSS Classes**

  Pseudo-elements can be combined with CSS classes:

```
p.article:first-letter {color:#ff0000;}
<p class="article">A paragraph in an article</p>
```

  The example above will display the first letter of all paragraphs with **class="article", in red.**

- **Multiple Pseudo-elements**

  Several pseudo-elements can also be combined.

In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

```
p:first-letter
{
color:#ff0000;
font-size:xx-large;
}
p:first-line
{
color:#0000ff;
font-variant:small-caps;
}
```

- **CSS - The :before Pseudo-element**

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

The following example inserts an image before each <h1> element:

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

- **CSS - The :after Pseudo-element**

24. CSS Navigation Bar

- **Navigation Bars**

25. CSS Image Opacity / Transparency

The CSS3 property for transparency is **opacity**.

```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* For IE8 and earlier */
}
```

26. CSS Image Sprites

- **Image Sprites**

  An image sprite is a collection of images put into a single image.

  A web page with many images can take a long time to load and generates multiple server requests.

  Using image sprites will reduce the number of server requests and save bandwidth.

27. CSS Media Types

With media types a page can have one layout for screen, one for print, one for handheld devices, etc.

- **Media Types**

28. CSS Attribute Selectors

- **Style HTML Elements With Specific Attributes**

  It is possible to style HTML elements that have specific attributes, not just class and id.

- **Attribute Selector**

  The example below styles all elements with a title attribute:

```
[title]
{
color:blue;
}
```

- **Attribute and Value Selector**

  The example below styles all elements with title="W3Schools":

```
[title=W3Schools]
{
border:5px solid green;
}
```

- **Attribute and Value Selector - Multiple Values**

  The example below styles all elements with a title attribute that contains a specified value. This works even if the attribute has space separated values:

```
[title~=hello] { color:blue; }
```

The example below styles all elements with a lang attribute that contains a specified value. This works even if the attribute has hyphen (-) separated values:

```
[lang|=en] { color:blue; }
```

- **Styling Forms**

  The attribute selectors are particularly useful for **styling forms** without class or ID:

```
input[type="text"]
{
width:150px;
display:block;
margin-bottom:10px;
background-color:yellow;
}
input[type="button"]
{
width:120px;
margin-left:35px;
display:block;
}
```

- **CSS Summary**

  This tutorial has taught you how to create style sheets to control the style and layout of multiple web sites at once.

  You have learned how to use CSS to add backgrounds, format text, add and format borders, and specify padding and margins of elements.

  You have also learned how to position an element, control the visibility and size of an element, set the shape of an element, place an element behind another, and to add special effects to some selectors, like links.

  The next step is to learn JavaScript.