

Web Design and Programming (0107558)

Internet Programming (0107571)¹

Cascading Style Sheets

Dr. Naeem Odat



College of Engineering
Department of Computer and Communications Engineering

¹Michael Mendez. *The Missing Link: An Introduction to Web Development and Programming*. Open SUNY Textbooks 2014.

WHAT IS CSS?

CSS

Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents.

Power of CSS

CSS ZEN GARDEN

Have a look at "<http://www.csszengarden.com/>" to appreciate the power of css.

Rule structure

What, Where, and When aspects

- ▶ **What:** the actual change we want to see (text color, line width, etc).
- ▶ **Where:** one or more specific locations, defined as a particular named element or when certain conditions are met (`p.before{}`).
- ▶ **When:** to control the site's appearance based on different factors like if the display is mobile sized, or if the user clicked the print button (`@media print {}`).

Rule structure

Where to place the rules

(1-rules-syntax.html)

- ▶ Inside `<style>` tag in `<head>`.

```
<style type="text/css">
  div, ul{                                /*--- Selectors      ---*/
    line-height: 1.8em;                  /*--- Declaration Property: value; ---*/
    background: darkgreen;               /*--- Declaration Property: value; ---*/
  }
</style>
```

- ▶ In a separate file.

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

- ▶ Inline. Inserting the rule inside the attribute tag of the item in question.

```
<p style="color: blue">This paragraph is blue because of its style</p>
```

Rule structure

Rule structure

```
<style type="text/css">
  div, ul{                               /*--- Selectors      ---*/
    line-height: 1.8em;                 /*--- Declaration Property: value; ---*/
    background: darkgreen;              /*--- Declaration Property: value; ---*/
  }
</style>
```

- ▶ Selector: an existing tag (unordered lists, paragraphs, inputs, etc), a custom class, or ID.
- ▶ Property: to adjust, and its value.
- ▶ A collection of groups of selectors-rules is called a style sheet.

Element selectors

```
p{  
    font-size:3em;  
}
```

Rule structure

Class selectors

- ▶ Used inside some tags. `class="myClass"`
- ▶ This allows us to apply different styles to the same selector type, while adding the ability to be more specific about which occurrence we are talking about.
- ▶ Defined with a period.

```
.myClass{  
    color:red;  
    background:gray;  
}
```


Rule structure

ID selectors

- ▶ Used inside a tag. `id="uniqueID"`.
- ▶ Defined with a `#`.
- ▶ An ID is used only once inside a page.

```
#uniqueID{  
    color:#00FFAD;  
    background: green;  
}
```

Rule structure

Grouping selectors

```
div, .myclass{  
    color:blue;  
}
```

Rule structure

Combining selectors

(3-combinedrules.html)

(Element with class selector (selector.class))
`p.myclass{ /*every p that has a class of myclass*/
 font-size:120px;
}`

(Child, direct, selector (selector>selector))
`div>span{ /*every span that is a direct child of div */
 color:red;
}`

(Descendant selector (selector selector))
`div span{ /*every descendant span from div */
 color:red;
}`

(Adjacent selectors (selector+selector))
`ol+ul{ /*at the same level and follow each other.*/
 color:green;
}`

Rule structure

Example

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title></title>
5      <style type="text/css">
6          #uniqueID{
7              color: blue;
8              background: yellow;
9          }
10     </style>
11     <style type="text/css">
12         .completed{
13             color: red;
14             background: gray;
15         }
16     </style>
17 </head>
18 <body>
19     <div class="completed">
20         This is a div content.<br>It should be green.
21     </div>
22     <div id="uniqueID">
23         Another div with some content<br>It should
24         be red.
25     </div>
26 </body>
</html>
```

This is a div content.

It should be green.

Another div with some content

It should be red.

Rule structure

Universal selector

*****: is the universal selector. This selector properties applied to all items.
footer*: applies to all items in footer.

Pseudo-class selectors

(4-sudoselectors.htm)

Pseudo classes. Keywords that may be added to the end of a selector to style an element when it's in a unique state.

- ▶ :link. A normal unvisited element
- ▶ :visited. An element that has been visited.
- ▶ :hover. Activated by mouse.
- ▶ :focus. Activated by the keyboard.
- ▶ :nth-child().
- ▶ :not(.completed). An element that doesn't have a completed class attribute.

Rule structure

Cascading part of CSS3

Browsers look at rules in the following order:

1. Browsers default.
2. External sheets.
3. Internal sheets.
4. Inline style.

Number 4 is going to overwrite number 3, number 3 overwrite number 2, and so on.

Rule structure

Attribute selectors

(5-attributes.htm)

- ▶ `=`: Match exactly `a[href='info.html']`
- ▶ `^`: Match the beginning exactly `a[href^='http://umich']`
- ▶ `$`: Match the end exactly `img[src$ = '.png']`, apply to .png images
- ▶ `*`: Wildcard `a[href*='umich']` has umich anywhere.

Rule structure

Conflict resolution

Origin precedence:

- ▶ When in conflict:

Last declaration wins.

- ▶ When no conflict:

Declarations merge.

Rule structure

Conflict resolution

Inheritance:

Inside elements in DOM inherit declarations from parent.

Rule structure

Conflict resolution

Specificity:

Most specific selector combination wins.

Score

style="..."	ID	Class, pseudo-class, attribute	# of Elements
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Example

```
div span{  
  color:blue;  
}
```

0	0	0	2
---	---	---	---

Rule structure

Example 1

```
div #myParag{  
  color:blue;  
}
```

```
div.big p{  
  color:red;  
}
```



Rule structure

Example 2

```
header.navigation p{  
    color:blue;  
}
```

```
p.blurb{  
    color:green;  
}
```

0	0	1	2		0	0	1	1
---	---	---	---	--	---	---	---	---

Conflict resolution

!important: when used for any rule it overrides all other rules.

```
p{  
    color:green !important;  
}
```

Rule structure

Order of Precedence

(6-precedence.html)

- The more specific the rule is, the greater influence it will have.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <style type="text/css">
    a{
      color:red;
    }
    div.block a{
      color: blue;
    }
  </style>
</head>
<body>
  <a href="">Our RED link</a>
  <div class="block">
    <a href="">Our BLUE link</a><br/>
    <a href="" style="color:green">Our GREEN link</a>
  </div>
</body>
</html>
```



Rule structure

Order of Precedence

(7-precedence.html)

- ▶ When rules have the same specificity the closer rule will override others.

```
<html>
<head>
  <title></title>
  <style type="text/css">
    a{
      color: blue;
      text-decoration: none;
    }
    a{
      color: red;
    }
  </style>
</head>
<body>
  <a href="">Our RED link</a>
</body>
</html>
```

Our RED link

Rule structure

Display

(8-display1.html)

- ▶ Exactly how elements are displayed—as block-level elements, inline elements, or something else—is determined by the display property.
- ▶ Every element has a default display property value; however, as with all other property values, that value may be overwritten.
- ▶ There are quite a few values for the display property, but the most common are:
 - ▶ **block**. Makes the element a block-level element.
 - ▶ **inline**. Makes the element an inline-level element.
 - ▶ **inline-block**. Allows an element to behave as a block-level element. However, the element will be displayed in line with other elements, and it will not begin on a new line by default.
 - ▶ **none**. Completely hides an element and renders the page as if that element doesn't exist. Any elements nested within this element will also be hidden.

Rule structure

More display values

- ▶ table.
- ▶ grid.
- ▶ flex.

Rule structure

display:table

(9-display2.html)

Used when you want your page layout to be arranged as in table without using HTML table tag.

You need to add "display:table" for the container where your elements should be cells. In each element a "display:table-cell" is added.

```
body{
  display:table;
}
div{
  width: 30%;
  height: 100px;
  background: #00ffff;
  display: table-cell;
}
```

Div A: is the one that resembles the first cell in the table. However, this is only a sample for using display value of table cell. In the upper level element the display value should be a table because this is a good programming practice to do.	Div B: it is also the one that resembles the first cell in the table. However, this is only a sample for using display value of table cell. In the upper level element the display value should be a table because this is a good programming practice to do.	Div C: one more is this one that resembles the first cell in the table. However, this is only a sample for using display value of table cell. In the upper level element the display value should be a table because this is a good programming practice to do.
---	---	---

```
<body>
  <div>Div A: is the one that resembles the first cell in the table.
  However, this is only a sample for using display value of table cell. In
  the upper level element the display value should be a table because this
  is a good programming practice to do.</div>
  <div>Div B: it is also the one that resembles the first cell in the
  table. However, this is only a sample for using display value of table
  cell. In the upper level element the display value should be a table
  because this is a good programming practice to do.</div>
  <div>Div C: one more is this one that resembles the first cell in the
  table. However, this is only a sample for using display value of table
  cell. In the upper level element the display value should be a table
  because this is a good programming practice to do.</div>
</body>
```

Rule structure

Overflow

(10-overflow.html)

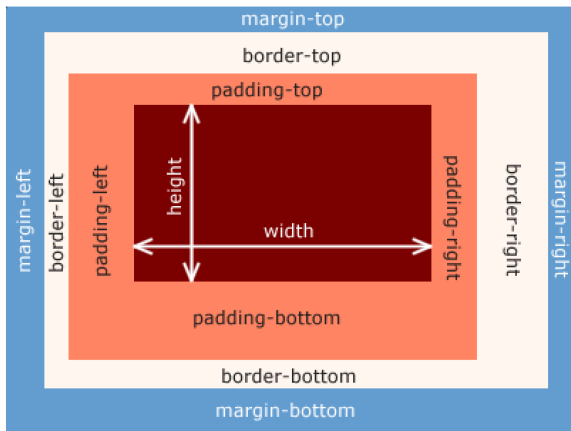
Has 4 values:

- ▶ visible: overflowed out of its container shown over other elements.
- ▶ hidden: text that doesn't fit inside is not shown.
- ▶ scroll: scroll bars appears to help shown text that is not shown.
- ▶ auto: scroll bars appear when needed.

Rule structure

Box Model

According to the box model concept, **every element** on a page is a rectangular box and may have width, height, padding, borders, and margins.



Rule structure

Box Model

- ▶ The core of the box is defined by the width and height of an element, which may be determined by:
 - ▶ the display property.
 - ▶ the contents of the element.
 - ▶ specified width and height properties.

```
div{  
  width: 50px;  
  height: 50px;  
  padding: 20px;  
  margin: 10px;  
  border: 10px solid red;  
}
```

Actual width=130px
Actual height=

Rule structure

Box Model

- ▶ The default margins and padding for text-based elements may differ from browser to browser and element to element.
- ▶ Use a CSS reset to tone all of these default values down to zero. Doing so allows us to work from the ground up and to specify our own values.

Rule structure

Margin and padding properties

- ▶ The margin and padding properties come in both longhand (each side alone) and shorthand form (e.g., `margin: 10px 20px 0 15px;`) (top, right, bottom, left).
- ▶ The margin and padding properties are completely transparent and do not accept any color values.

Rule structure

Border property

(11-boxmodel.html)

- ▶ The border property requires three values: width, style, and color (e.g., `border: 6px solid #ff0000;`).
- ▶ The most common style values for border appearance are solid, double, dashed, dotted, and none.
- ▶ Borders can be placed on one side of an element at a time if we'd like (e.g., `border-bottom: 6px solid #949599;`).
- ▶ `border-radius` property, enables us to round the corners of an element (e.g., `border-radius: 5px; border-radius: 15px 75px;`).

Rule structure

Transitions

When elements transition from one state to another, you can alter their appearance, like hovering over a div.

The Properties

- ▶ transition-property. What is it you want to change? (size, color, position, etc.)
- ▶ transition-duration. How long should each transition last?
- ▶ transition-timing. Should it be a smooth transition (linear)? Or different?
- ▶ transition-delay. How long should the wait be before the transition begins?

Rule structure

Transitions

CSS

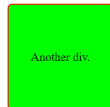
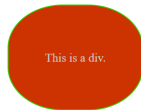
```
div{  
  width: 100px;  
  height: 100px;  
  line-height: 100px;  
  padding: 20px;  
  margin: 30px;  
  background: #00ff00;  
  border: 1px solid #ff0000;  
  border-radius: 6px;  
  text-align: center;  
  transition-property: color,width,border-radius, background, border;  
  transition-duration: .5s;  
  transition-timing-function: linear;  
  transition-delay: .5s;  
}  
div:hover{  
  color: #ffffff;  
  width: 150px;  
  background: #ff0000;  
  border-radius: 50%;  
  border: 1px solid #00ff00;  
}
```

(12-transition.html)

HTML

```
<div> This is a div.</div>  
<div>Another div.</div>
```

Result



Rule structure

Transforms

- ▶ This property allows you to rotate, scale, move, skew, etc., elements.
- ▶ The transform property applies a 2D or 3D transformation to an element.

Rule structure

Transforms

- ▶ transform: rotate(angle in deg)
- ▶ transform: scale(x,y)
- ▶ transform: skew(x in deg, y in deg)

(13-transform.html)


CSS

```
div.a{
  width: 100px;
  height: 50px;
  margin: 5px;
  line-height: 50px;
  text-align: center;
  border-radius: 2px;
  background: #deaddead;
  border: 2px solid #ff0000;
  transform: rotate(30deg);
}
div.b{
  width: 100px;
  height: 50px;
  margin: 5px;
  line-height: 50px;
  text-align: center;
  border-radius: 2px;
  background: #ceedceed;
  border: 2px solid #00ff00;
  transform: scale(1.5,1);
}
div.c{
  width: 100px;
  height: 50px;
  margin: 5px;
  line-height: 50px;
  text-align: center;
  border-radius: 2px;
  background: #feedfeed;
  border: 2px solid #0000ff;
  transform: skew(30deg,0deg);
}
```

HTML

```
<div class="a">Some text</div>
<div class="b">Text</div>
<div class="c">Another text.</div>
```

Result

Before	After
	
	
	

Rule structure

Positioning

- ▶ The position property specifies the type of positioning method used for an element:
 - ▶ static. Default value for all elements (except html it is not static, it is relative). Not affected by top, right...
 - ▶ relative. Positioned relative to its normal position as given by top, etc.
 - ▶ fixed. Stayed in the same place relative to the view port even if the page is scrolled.
 - ▶ absolute. Positioned relative to the nearest positioned ancestor.
 - ▶ sticky. Positioned based on the user's scroll position.
- ▶ Elements are then positioned using the top, bottom, left, and right properties

Rule structure

Positioning - Demo

(14-positioning.html)