Web Programming

YJ – 2016

CSS

CSS exists to style your HTML - to present your content. The language of Cascading Style Sheets is somewhat different to HTML but it remains simple and straightforward.

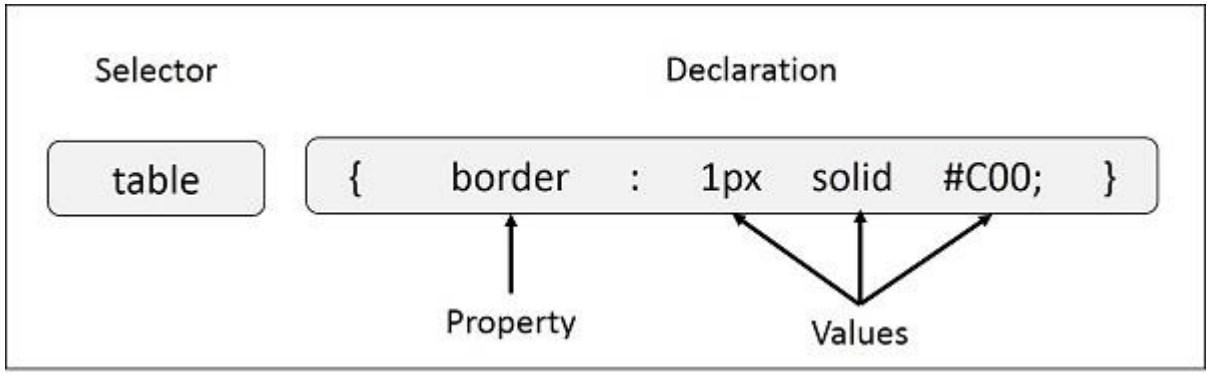
```
body {
    margin:0 auto;
    padding:0;
    font-family: Arial, Helvetica, sans-serif;
    font-size:100%;
}

a {
    color:#0e61cc;
    text-decoration:none;
}

a:hover {
    color:#0e61cc;
    text-decoration:underline;
}

div#bg-wrapper {
    width:1000px;
    margin:0 auto;
    background:url(images_new/body-bg.gif) center top repeat-y;
}

div#wrapper {
    width:970px;
    text-align:center;
    margin:0 auto;
}

/* ############## HEADER ######### */
div#header {
    height:62px;
    background:#15447e url(images_new/logo.jpg) top left no-repeat;
}
```

CSS Syntax

- **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 value either *red* or *#F1F1F1* etc.



CSS type

There are three ways to apply CSS to HTML: In-line, internal, and external

- In-line

```
<p style="color: red">text</p>
```

- Internal

```
<html>
<head>
<title>CSS Example</title>
<style>
    p {
        color: red;
    }
    a {
        color: blue;
    }
</style>
```

- External

```
<head>
    <title>CSS Example</title>
    <link rel="stylesheet"
href="style.css">
</head>
```

Selectors

- Universal selectors

```
* {
    margin: 0;
    padding: 0;
}

#contact * {
    display: block;
}
```

- Child selectors

```
#genus_examples > li { border: 1px solid red }
```

- Adjacent selectors

```
h1 + p { font-weight: bold }
```

Selectors

The Type Selectors

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

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;
}
```

The Descendant Selectors

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element.

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

Selectors

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.

```
h1.black {
   color: #000000;
}
```

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

Selectors

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.

```
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;
}
```

Selectors

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 direct child of <body> element. Other paragraphs put inside other elements like or <td> would not have any effect of this rule.

Attribute Selectors

- `input[type=text] { width: 200px; }`
- `input[type=text][disabled] { border: 1px solid #ccc }`

- `[attribute^=something]` will match a the value of an attribute that begins with something.
- `[attribute$=something]` will match a the value of an attribute that ends with something.
- `[attribute*=something]` will match a the value of an attribute that contains something, be it in the beginning, middle, or end.

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;
}
```

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.

Class & ID

- In the CSS, a class selector is a name preceded by a full stop (“.”) and an ID selector is a name preceded by a hash character (“#”).

```
#top {
    background-color: #ccc;
    padding: 20px
}

.intro {
    color: red;
    font-weight: bold;
}
/////////////////////////////////////
<div id="top">

<h1>Chocolate curry</h1>

<p class="intro">This is my recipe for making curry purely with chocolate</p>

<p class="intro">Mmm mm mmmmm</p>

</div>
```

Grouping

```
h2 {
    color: red;
}

.thisOtherClass {
    color: red;
}

.yetAnotherClass {
    color: red;
}
```

Equals

```
h2, .thisOtherClass,
.yetAnotherClass {
    color: red;
}
```

Nesting

```
#top {
    background-color: #ccc;
    padding: 1em
}

#top h1 {
    color: #ff0;
}

#top p {
    color: red;
    font-weight: bold;
}

////////////////////////////
<div id="top">
    <h1>Chocolate curry</h1>
    <p>This is my recipe for making curry
purely with chocolate</p>
    <p>Mmm mm mmmmm</p>
</div>
```

Measurement Units

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;}
rem	A relative measurement to base font size for the height of a font in em spaces.	p {letter-spacing: 7rem;}
px	Defines a measurement in screen pixels.	p {padding: 25px;}

Colors

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

Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

Background

- background-color, which we have come across before.
- background-image, which is the location of the image itself.
- background-repeat, which is how the image repeats itself. Its value can be:
  - repeat, the equivalent of a “tile” effect across the whole background,
  - repeat-y, repeating on the y-axis, above and below,
  - repeat-x (repeating on the x-axis, side-by-side), or
  - no-repeat (which shows just one instance of the image).
- background-position, which can be top, center, bottom, left, right, a length, or a percentage, or any sensible combination, such as top right.

Background

```
<html>
   <head>
      <style>
         body  {
            background-image: url("/css/images/css.jpg");
            background-color: #cccccc;
             background-repeat: repeat;
             background-position:100px 200px;



         }
      </style>
      <body>
         <h1>Hello World!</h1>
      </body>
   </head>
<html>
```

Background

Multiple backgrounds

```
background: url(bg.png), url(bullet.png) 0 50% no-repeat,
url(arrow.png) right no-repeat;
```

Gradients

```
background: linear-gradient(orange, red);

background: linear-gradient(to bottom right, orange, red);

background: linear-gradient(20deg, orange, red);

background: linear-
gradient(hsl(0,100%,50%),hsl(60,100%,50%),hsl(120,100%,50%
),hsl(180,100%,50%),hsl(240,100%,50%),hsl(300,100%,50%));
```

Background

Radial gradients

```
background: radial-gradient(yellow, green);
background: radial-gradient(circle closest-side, yellow, green);
background: radial-gradient(at top left, yellow, green);
```

Repeating gradients

```
background: repeating-linear-gradient(white, black 10px, white 20px);
```

Text

- Font-family
  - E.g. “Arial”
- Font-size
- Font-weight
  - Normal, bold, bolder, lighter, 100, 200, 300, 400 (same as normal), 500, 600, 700 (same as bold), 800 or 900
- Font-style
  - Italic capitalize, uppercase, lowercase, none

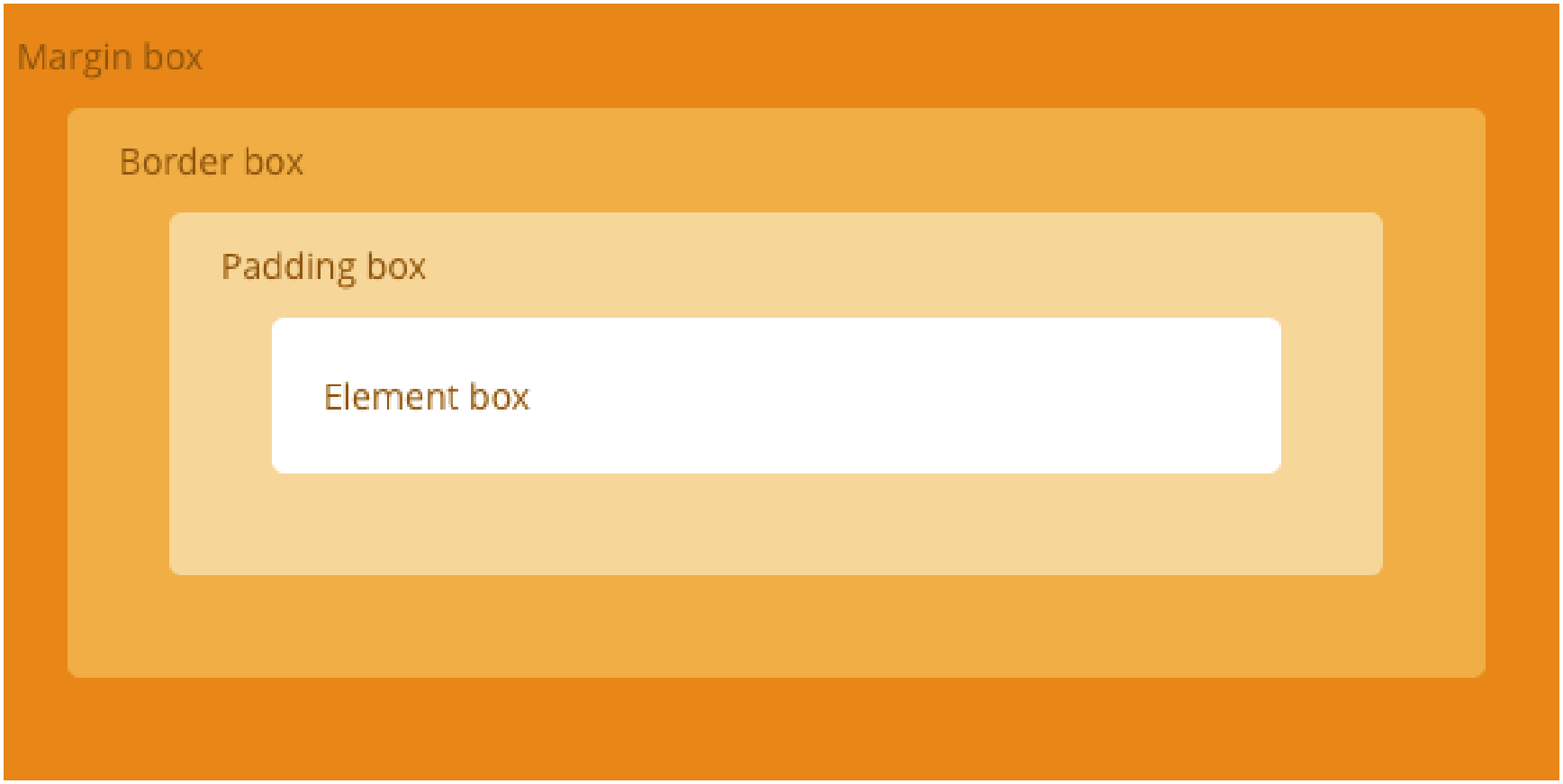
- The color property is used to set the color of a text.
- The direction property is used to set the text direction.
- The letter-spacing property is used to add or subtract space between the letters that make up a word.
- The word-spacing property is used to add or subtract space between the words of a sentence.
- The text-indent property is used to indent the text of a paragraph.
- The text-align property is used to align the text of a document.
- The text-decoration property is used to underline, overline, and strikethrough text.
- The text-transform property is used to capitalize text or convert text to uppercase or lowercase letters.
- The white-space property is used to control the flow and formatting of text.
- The text-shadow property is used to set the text shadow around a text.

Borders

The border property allows you to specify color, style, and width of lines in one property –

```
<html>
   <head>
   </head>
   <body>
      <p style="border:4px solid red;">
      This example is showing shorthand property for border.
      </p>
   </body>
</html>
```

Margin & Padding



Margin & Padding

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

```
h2 {
    font-size: 1.5em;
    background-color: #ccc;
    margin: 20px;
    padding: 40px auto;
}
```

Dimension

- The height property is used to set the height of a box.
- The width property is used to set the width of a box.
- The line-height property is used to set the height of a line of text.
- The max-height property is used to set a maximum height that a box can be.
- The min-height property is used to set the minimum height that a box can be.
- The max-width property is used to set the maximum width that a box can be.
- The min-width property is used to set the minimum width that a box can be.

```
<html>
   <head>
   </head>
   <body>
      <p style="width:400px; height:100px; border:1px solid red; padding:5px;
margin:10px;">
      This paragraph is 400pixels wide and 100 pixels high
   </p>
   </body>
</html>
```

Visibility

Value	Description
visible	The box and its contents are shown to the user.
hidden	The box and its content are made invisible, although they still affect the layout of the page.
collapse	This is for use only with dynamic table columns and row effects.

```
<html>
   <head>
   </head>
   <body>
      <p>
      This paragraph should be visible in normal way.
      </p>

      <p style="visibility:hidden;">
      This paragraph should not be visible.
      </p>
   </body>
</html>
```

Layers

A z-index property can help you to create more complex webpage layouts. Following is the example which shows how to create layers in CSS.

```
<html>
   <head>
   </head>
   <body>
      <div style="background-color:red; width:300px; height:100px;
position:relative; top:10px; left:80px; z-index:2">
      </div>

      <div style="background-color:yellow; width:300px; height:100px;
position:relative; top:-60px; left:35px; z-index:1;">
      </div>

      <div style="background-color:green; width:300px; height:100px;
position:relative; top:-220px; left:120px; z-index:3;">
      </div>
   </body>
</html>
```

Pseudo Classes

- Pseudo classes are bolted on to selectors to specify a state or relation to the selector. They take the form of selector:pseudo_class { property: value; }, simply with a colon in between the selector and the pseudo class.

```
a:link {
    color: blue;
}
a:visited {
    color: purple;
}
a:active {
    color: red;
}
a:hover {
    text-decoration: none;
    color: blue;
    background-color: yellow;
}

input:focus, textarea:focus {
    background: #eee;
}
p:first-child {
    font-weight: bold;
    font-size: 40px;
}
```

Links

```
<html>
   <head>
<style type="text/css">
   a:link {color: #000000}
   a:visited {color: #006600}
   a:hover {color: #FFCC00}
   a:active {color: #FF00CC}
   a:focus {color: #0000FF}

</style>
   </head>
   <body>
      <a href="">Link</a>
   </body>
</html>
```

Pseudo Elements

```
p {
    font-size: 12px;
}

p:first-letter {
    font-size: 24px;
    float: left;
}

p:first-line {
    font-weight: bold;
}

li:before {
    content: "POW! ";
}

P:after {
    content: url(images/jam.jpg);
}
```

Shorthand Properties

```
p {
    font: 14px/1.5 "Times New Roman", times, serif;
    padding: 30px 10px;
    border: 1px black solid;
    border-width: 1px 5px 5px 1px;
    border-color: red green blue yellow;
    margin: 10px 50px;
}
```

Positioning

- static is the default value and renders a box in the normal order of things, as they appear in the HTML.
- relative is much like static but the box can be offset from its original position with the properties top, right, bottom and left.
- absolute pulls a box out of the normal flow of the HTML and delivers it to a world all of its own. In this crazy little world, the absolute box can be placed anywhere on the page using top, right, bottom and left.
- fixed behaves like absolute, but it will absolutely position a box in reference to the browser window as opposed to the web page, so fixed boxes should stay exactly where they are on the screen even when the page is scrolled.

Floating

- Floating a box will shift it to the right or left of a line, with surrounding content flowing around it.
- Float:Left
- Float:right
- Float:center

- clear: left will clear left floated boxes
- clear: right will clear right floated boxes
- clear: both will clear both left and right floated boxes.

Border Radius

- `border-radius: 10px;`

- Multiple Value
- `border-radius: 3px 6px 9px 12px;`

- Ellispses
- `border-radius: 50px/100px;`
- `border-bottom-left-radius: 50px;`
- `border-bottom-right-radius: 50px;`

Shadows

- `box-shadow: 5px 5px 3px 1px #999`
- `box-shadow: inset 0 0 7px 5px #ddd;`
- `text-shadow: -2px 2px 2px #999;`
  - The first value is the horizontal offset
  - The second value is the vertical offset
  - The third value is the blur radius (optional)
  - The fourth value is the color (optional, although omitting this will make the shadow the same color as the text itself)

Transition

```
a:link {
    transition: all .5s linear 0;
    color: #ff0000;
}
a:hover {
    color: #ffff00;
}

transition: .5s;
transition-property: color, font-size;

transition: color .5s, font-size 2s;
```

Rules

Importing

```
@import url(morestyles.css);
```

Media

```
@media print {
    body {
        font-size: 10pt;
        font-family: times, serif;
    }
}
@media screen, projection {
    /* … */
}
```

Rules

Screen size

```
@media screen and (max-width: 1000px) {
    #content { width: 100% }
}

@media screen and (max-width: 800px) {
    #nav { float: none }
}

@media screen and (max-width: 600px) {
    #content aside {
        float: none;
        display: block;
    }
}
```

Rules

Orientation

```
@media screen and (orientation: landscape) {
    #nav { float: left }
}

@media screen and (orientation: portrait) {
    #nav { float: none }
}
```

Device

```
@media screen and (min-device-height: 768px) and (max-device-width: 1024px) {
    /* You can apply numerous conditions separated by "and" */
}
@media screen and (resolution: 326dpi) { /* */ }

@media screen and (min-resolution: 96dpi) { /* */ }

@media screen and (device-aspect-ratio: 16/9) { /* */ }
```