

# Lecture 4

**CSS**



**Intro to CSS**

# Topics

- Box Model
- CSS Units
- CSS Colors
- Borders
- Style and Image
- Style the Font
- CSS Stylesheets
- [Link to StyleSheet](#)

# CSS Units

# CSS Units



- CSS has several different units for expressing a length.
- Many CSS properties take "**length**" values, such as **width**, **margin**, **padding**, **font-size**, etc.
- Length is a number followed by a **length unit**, such as 10px, 2em, etc.
- A whitespace cannot appear between the number and the unit. However, if the value is 0, the unit can be omitted.
- For some CSS properties, negative lengths are allowed.
- There are two types of length units: **absolute** and **relative**.

# Numbers, Lengths and Percentages

- There are various numeric data types that you might find yourself using in CSS. The following are all classed as numeric:

Data type	Description
<code>&lt;integer&gt;</code>	An <code>&lt;integer&gt;</code> is a whole number such as <code>1024</code> or <code>-55</code> .
<code>&lt;number&gt;</code>	A <code>&lt;number&gt;</code> represents a decimal number — it may or may not have a decimal point with a fractional component. For example, <code>0.255</code> , <code>128</code> , or <code>-1.2</code> .
<code>&lt;dimension&gt;</code>	A <code>&lt;dimension&gt;</code> is a <code>&lt;number&gt;</code> with a unit attached to it. For example, <code>45deg</code> , <code>5s</code> , or <code>10px</code> . <code>&lt;dimension&gt;</code> is an umbrella category that includes the <code>&lt;length&gt;</code> , <code>&lt;angle&gt;</code> , <code>&lt;time&gt;</code> , and <code>&lt;resolution&gt;</code> types.
<code>&lt;percentage&gt;</code>	A <code>&lt;percentage&gt;</code> represents a fraction of some other value. For example, <code>50%</code> . Percentage values are always relative to another quantity. For example, an element's length is relative to its parent element's length.

# Absolute Lengths

```
h1 {font-size: 1.5cm;}
h2 {font-size: 1cm;}
p {
  font-size: 0.5cm;
  line-height: 1cm;
}
```

```
h1 {font-size: 50px;}
h2 {font-size: 30px;}
p {
  font-size: 15px;
  line-height: 20px;
}
```

- The **absolute length units** are **fixed** and a length expressed in any of these will appear as exactly that size.
- Absolute length units are not recommended for use on screen, because screen sizes vary so much.
- However, they can be used if the output medium is known, such as for print layout.

# Absolute Lengths

- \* **Pixels (px)** are relative to the **viewing device**. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.

Unit	Name	Equivalent to
cm	Centimeters	1cm = 96px/2.54
mm	Millimeters	1mm = 1/10th of 1cm
Q	Quarter-millimeters	1Q = 1/40th of 1cm
in	Inches	1in = 2.54cm = 96px
pc	Picas	1pc = 1/6th of 1in
pt	Points	1pt = 1/72th of 1in
px	Pixels	1px = 1/96th of 1in

# Relative Lengths

```
p {  
  font-size: 16px;  
  line-height: 2em;  
}  
  
div {  
  font-size: 30px;  
  border: 1px solid black;  
}  
  
span {  
  font-size: 0.5em;  
}
```

- Relative length units specify a length relative to another length property.
- Relative length units scales better between different rendering mediums.
- The benefit of using relative units is that with some careful planning you can make it so the size of text or other element scales relative to everything else on the page.



# Relative Lengths

Unit	Relative to
em	Font size of the parent, in the case of typographical properties like <code>font-size</code> , and font size of the element itself, other properties like <code>width</code> .
ex	x-height of the element's font.
ch	The advance measure (width) of the glyph "0" of the element's font.
rem	Font size of the root element.
lh	Line height of the element.
vw	1% of the viewport's width.
vh	1% of the viewport's height.
vmin	1% of the viewport's smaller dimension.
vmax	1% of the viewport's larger dimension.

```
.wrapper {  
  font-size: 1em;  
}
```

```
.px {  
  width: 200px;  
}
```

```
.vw {  
  width: 10vw;  
}
```

```
.em {  
  width: 10em;  
}
```

```
<div class="wrapper">  
  <div class="box px">I am 200px wide</div>  
  <div class="box vw">I am 10vw wide</div>  
  <div class="box em">I am 10em wide</div>  
</div>
```

I am 200px wide

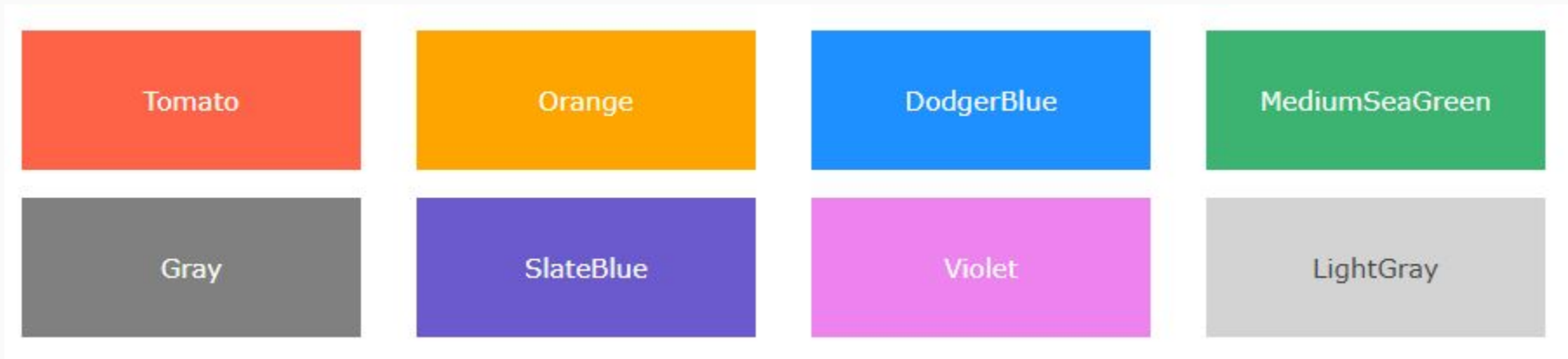
I  
am  
10vw  
wide

I am 10em wide

# CSS Colors

# CSS Color Names

- Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.
- CSS/HTML support [140 standard color names](#).
- In CSS, a color can be specified by using a color name:



# CSS Color Values

- In CSS, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values: Same as color name "Tomato":

`rgb(255, 99, 71)`

`#ff6347`

`hsl(9, 100%, 64%)`

# CSS Color Values

- Same as color name "Tomato", but 50% transparent:

```
rgba(255, 99, 71, 0.5)
```

```
hsla(9, 100%, 64%, 0.5)
```

# CSS Text Colors

- We can set the color of Text and Borders

## Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

## CSS Border Colors

- We can use CSS to control and set the color of the border of elements

Hello World

Hello World

Hello World

```
<h1 style="border:2px solid Tomato;">Hello World</h1>  
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>  
<h1 style="border:2px solid Violet;">Hello World</h1>
```



# CSS Backgrounds



# Background Colors

- The `background-color` property defines the background color on any element in CSS. The property accepts any valid `<color>`. A background-color extends underneath the content and padding box of the element.
- In the example below, we have used various color values to add a background color to the box, a

```
.box {  
  background-color: #567895;  
}  
  
h2 {  
  background-color: black;  
  color: white;  
}  
  
span {  
  background-color: rgba(255,255,255,.5);  
}
```

## Background Colors

Try changing the background `colors`.

```
<div class="box">  
  <h2>Background Colors</h2>  
  <p>Try changing the background <span>colors</span>.</p>  
</div>
```

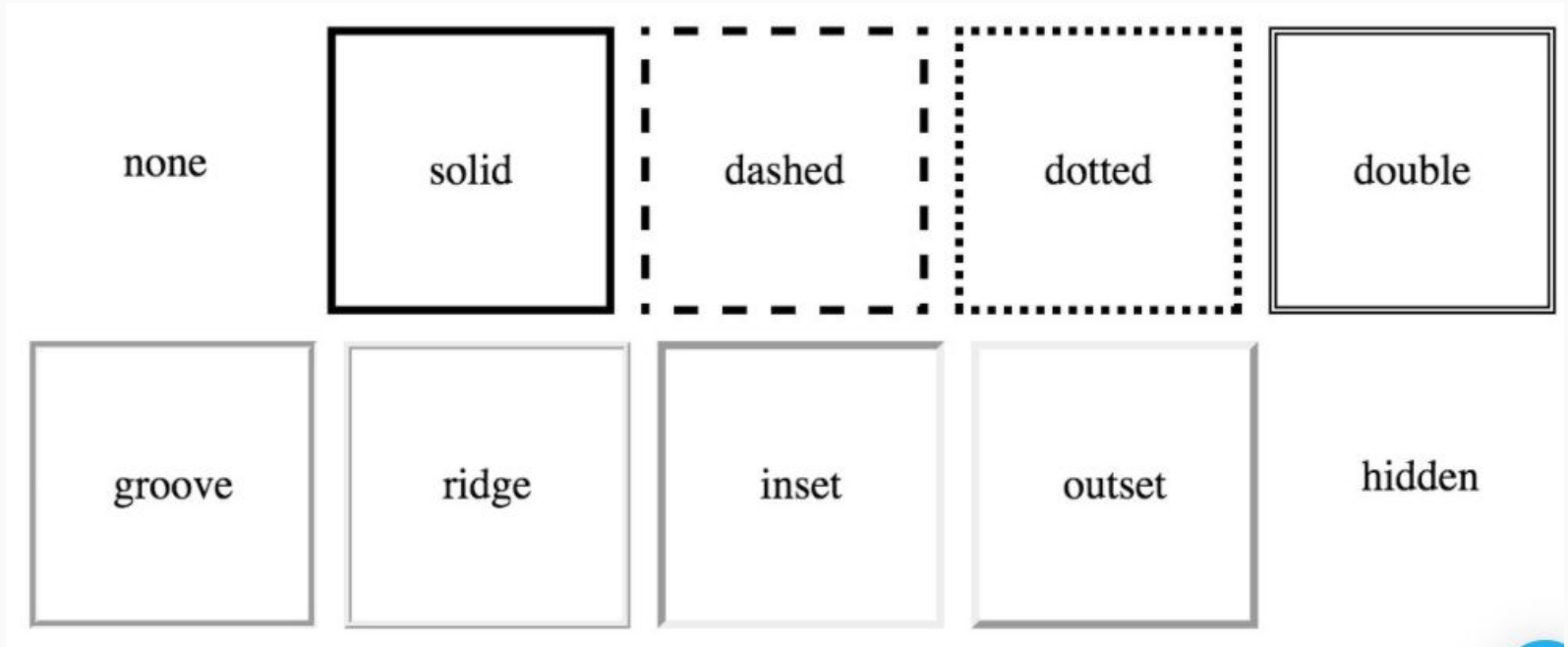


# Borders



# Borders

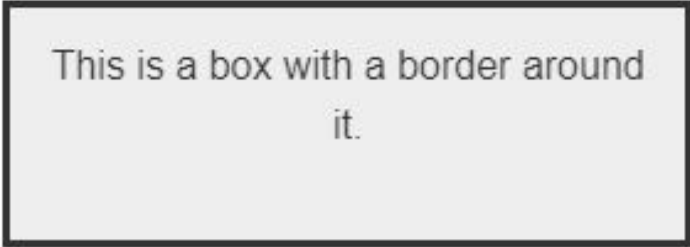
If you've ever used a table in a word processor or spreadsheet, then you should be familiar with borders. With CSS, you can add a border to just about anything. There are a ton of different options for customizing borders like style, width, and color!



# Borders

- The **border** shorthand CSS property sets an element's border.
- It sets the values of **border-width**, **border-style**, and **border-color**.

```
/* style */  
border: solid;  
  
/* width | style */  
border: 2px dotted;  
  
/* style | color */  
border: outset #f33;  
  
/* width | style | color */  
border: medium dashed green;
```



This is a box with a border around  
it.

# CSS Fonts

# CSS Fonts

The CSS font properties define the font family, boldness, size, and the style of a text.

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width



# CSS Font Family

- The font family of a text is set with the `font-family` property.
- The `font-family` property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

```
.serif {  
  font-family: "Times New Roman", Times, serif;  
}  
  
.sansserif {  
  font-family: Arial, Helvetica, sans-serif;  
}
```



Sans-serif



Serif



Serif  
(red serifs)

# Image Styles



# Resizing Images

So what can we do about the **overflowing issue**?

A common technique is to make the **max-width** of an image 100%. This will enable the image to become smaller in size than the box but not larger. This technique will work with other replaced elements such as **<video>**s.



```
.box {  
  width: 200px;  
  height: 200px;  
}  
  
img {  
  height: 100%;  
  width: 100%;  
}
```

```
<div class="wrapper">  
  <div class="box"></div>  
  <div class="box"></div>  
</div>
```

# Box Shadow

*Use shadows to add a sense of depth to images*

The **box-shadow** CSS property adds shadow effects around an element's frame. You can set multiple effects separated by commas. A box shadow is described by X and Y offsets relative to the element, blur and spread radius, and color.

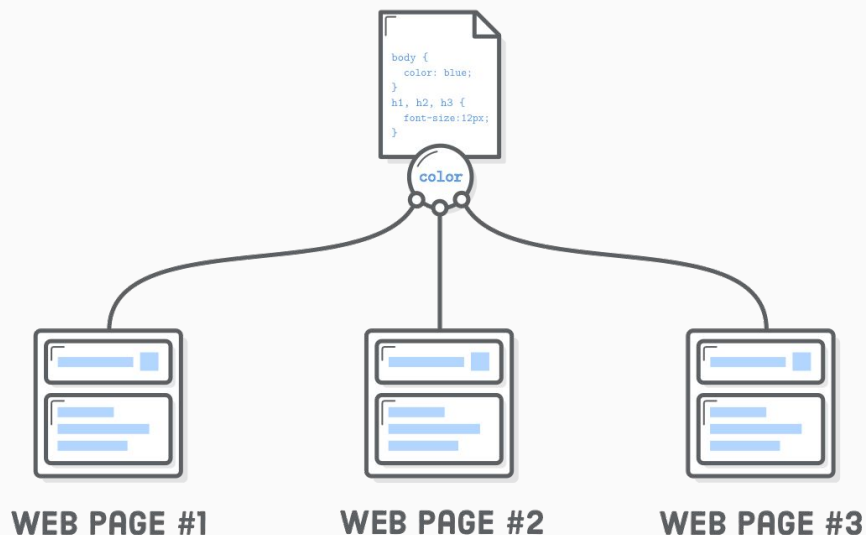


```
/* offset-x | offset-y | blur-radius | color */  
box-shadow: 10px 5px 5px black;
```

# Stylesheets

# What is a Stylesheet?

## GLOBAL CSS STYLES



*What if you wanted to use the same CSS on more than one webpage?*

You could just copy all of your CSS from one file and paste it into another, but that seems like a lot of extra work and doesn't scale very well. What if you decide to make changes later? You'd have to change every copy of the CSS!

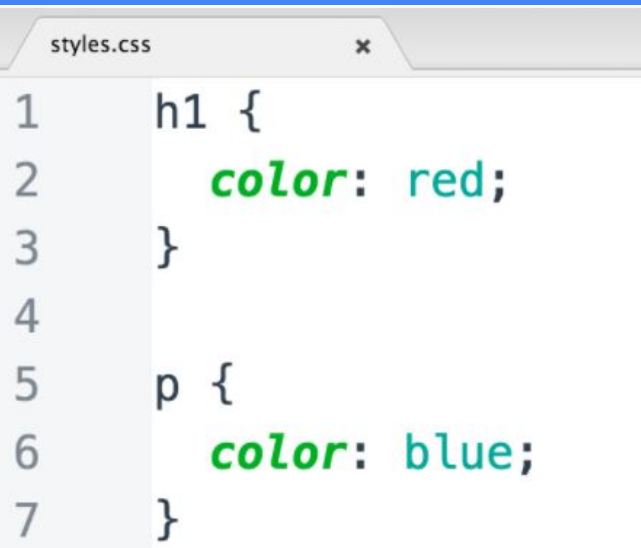
While the process described above works, it's not recommended. Instead, the preferred method is to write your CSS in a file called a **stylesheet** and then link to that file in your HTML.

# Stylesheets

A stylesheet is a file containing the code that describes how elements on your webpage should be displayed.

This is no different than what you've been doing before, except the CSS lives in a different file... and you don't have to use the **<style>** tags anymore.

To create a stylesheet, simply add a new file to your project, write some CSS, and save it as **name-of-stylesheet.css**.



```
styles.css x
1  h1 {
2      color: red;
3  }
4
5  p {
6      color: blue;
7  }
```



# How to Link to a Stylesheet

Before your webpage can use the stylesheet, you need to link to it. To do this, you'll need to create a `<link>` to your stylesheet in your HTML. To create a link, simply type the following inside the `<head>` of your HTML.

```
<link href="path-to-stylesheet/stylesheet.css" rel="stylesheet">
```

The `href` attribute specifies the **path** to the linked resource (you can link to other resources besides stylesheets, more on this later) and the `rel` attribute names the **relationship** between the resource and your document. In this case, the relationship is a stylesheet. When you're done, it will look something like this...

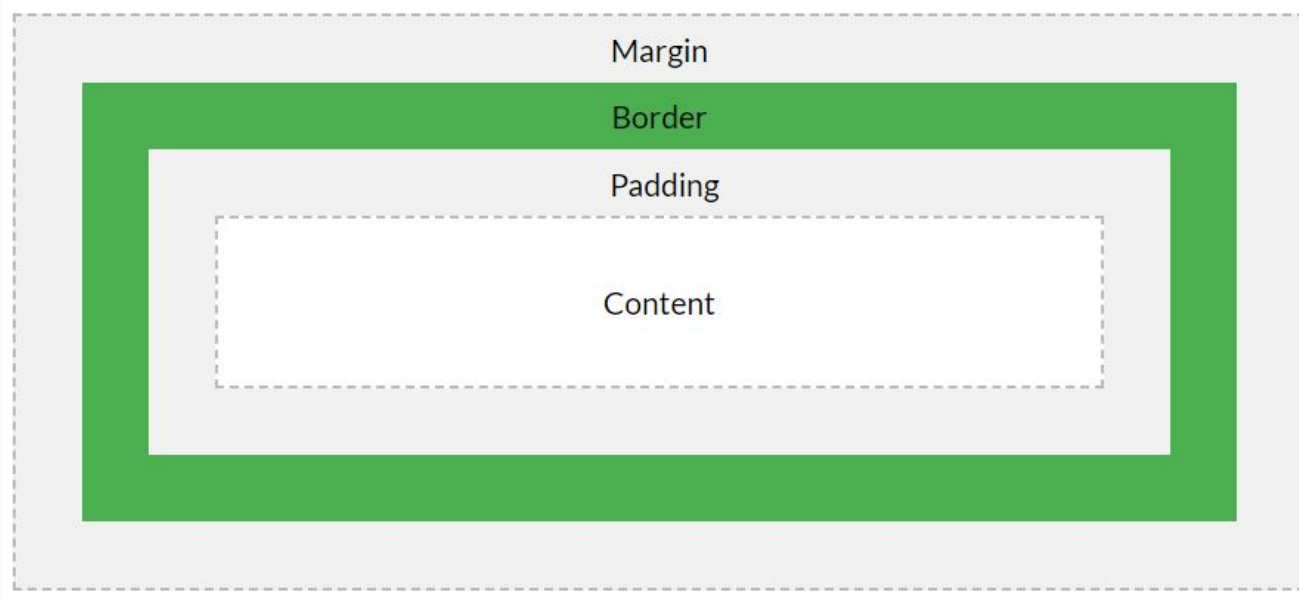
```
<head>
  <title>My Webpage</title>
  <!-- ... -->
  <link href="path-to-stylesheet/stylesheet.css" rel="stylesheet">
  <!-- ... -->
</head>
```

# Box Model

# The CSS Box Model

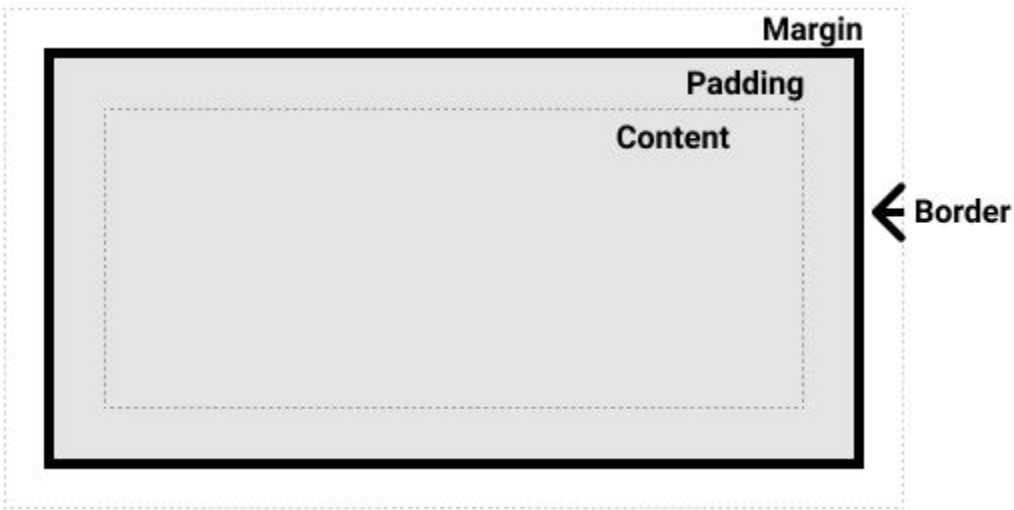
All HTML elements can be considered as boxes. In CSS, the term "**box model**" is used when talking about design and layout.

The CSS **box model** is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



# Parts of the Box

Making up a block box in CSS we have the:



- **Content box:** The area where your content is displayed, which can be sized using properties like [width](#) and [height](#).
- **Padding box:** The padding sits around the content as white space; its size can be controlled using [padding](#) and related properties.
- **Border box:** The border box wraps the content and any padding. Its size and style can be controlled using [border](#) and related properties.
- **Margin box:** The margin is the outermost layer, wrapping the content, padding and border as whitespace between this box and other elements. Its size can be controlled using [margin](#) and related properties.

# Using Dev Tools to view the box model

Your [browser developer tools](#) can make understanding the box model far easier. If you inspect an element in Firefox's DevTools, you can see the size of the element plus its margin, padding, and border. Inspecting an element in this way is a great way to find out if your box is really the size you think it is!

