

# WEEK 3 | PROGRAMMING

## CSS BOX MODEL

Shawn Park & Jeff Zhan



Web Design DeCal

DESIGN MEETS PROGRAMMING

# Review

- What are some CSS properties that we went over last week?
- Given the following HTML, how would you make the element with the text “Red” have a red background color and white text?

```
<div id="blue" class="colors">Blue</div>  
<div id="red" class="colors">Red</div>  
<div id="green" class="colors">Green</div>
```

```
#red {  
    background-color: red;  
    color: white;  
}
```



# *Review* CSS



# Today's Outline

1. CSS Box Model
2. CSS Color





***Goal Today:***  
*To Structure your Web Pages*



# CSS Box Model

# CSS Box Model

- The **CSS Box Model** is the standard way of structuring your elements and web pages. It allows us to add *space* and *borders* to an element





# CSS Box Model



*Facebook makes use of the Box Model everywhere*



# CSS Box Model

- 4 Key components
  - Content
  - Padding
  - Border
  - Margin
- **Content** is everything inside an element
  - Example: In `<p>Hello World</p>`, “Hello World” is the content
- **Border** is the line that surrounds an element
- **Padding** is the space between the content and border
- **Margin** is the space from the border and nearby elements



# CSS Box Model

- We went over border last week
- **border**
  - Takes 3 values: width, style, color
  - Example: *border: 1px dashed blue;*
- Padding and Margin can take in 1, 2, 3 or 4 values
- **1 Value:**
  - *padding: 10px;* = 10 pixels of space all around, from content to border
- **2 Values:**
  - *margin: 5px 10px;* = 5 pixels of space above and below border, 10 pixels of space to the left and right of border
  - Corresponds to top & bottom, left & right

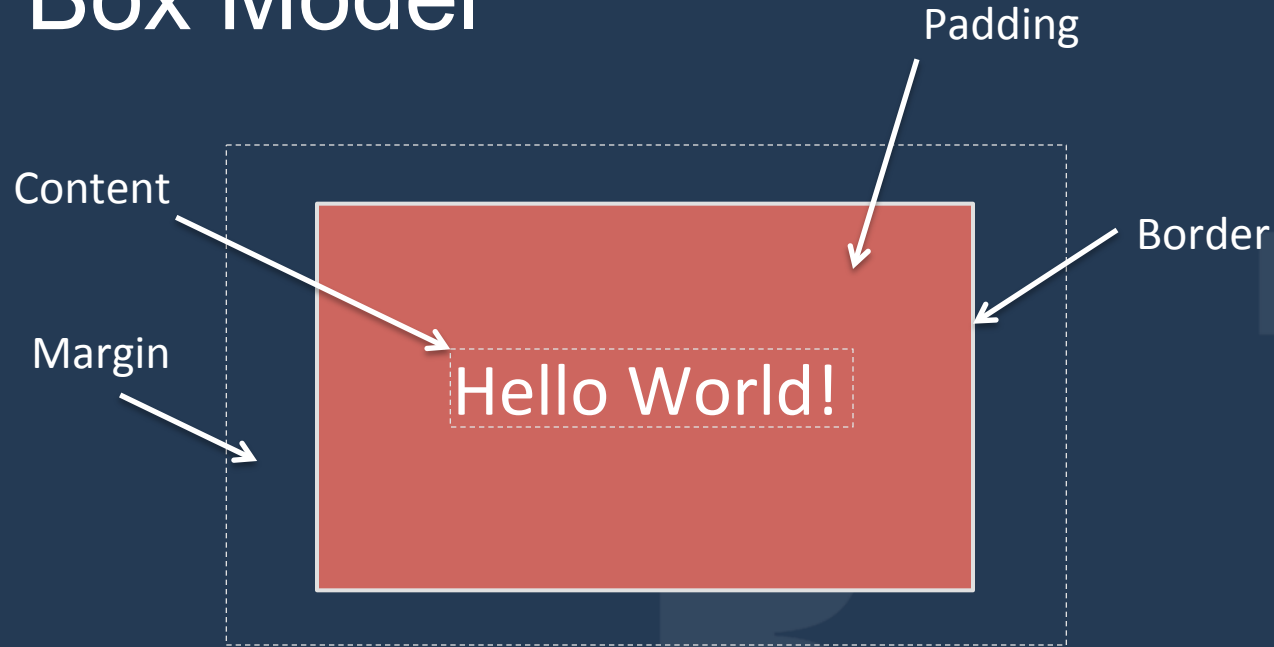




# CSS Box Model

- **3 Values:**
  - *padding: 10px 5px 20px; = Corresponds to top, left & right, bottom*
- **4 Values:**
  - *margin: 10px 5px 20px 15px; = Corresponds to top, right, bottom, left (clockwise)*

# CSS Box Model



- In this example, a possible style for the element is:
  - *padding: 15px;*
  - *border: 2px solid #ccc; /\* light-gray \*/*
  - *margin: 5px 10px;*

# CSS Box Model



- Also know that if you add a *background-color* to your image, it will color in the **content** and the **padding** portions of the element
- *Background-color* will not affect the **margin!!!**

# CSS Box Model

- Currently we have divs that stack vertically. What does this mean?
- **Divs** are, by default, *blocks*
  - They take up the entire row
  - By default, you **cannot** stack divs side by side
  - They look like this:



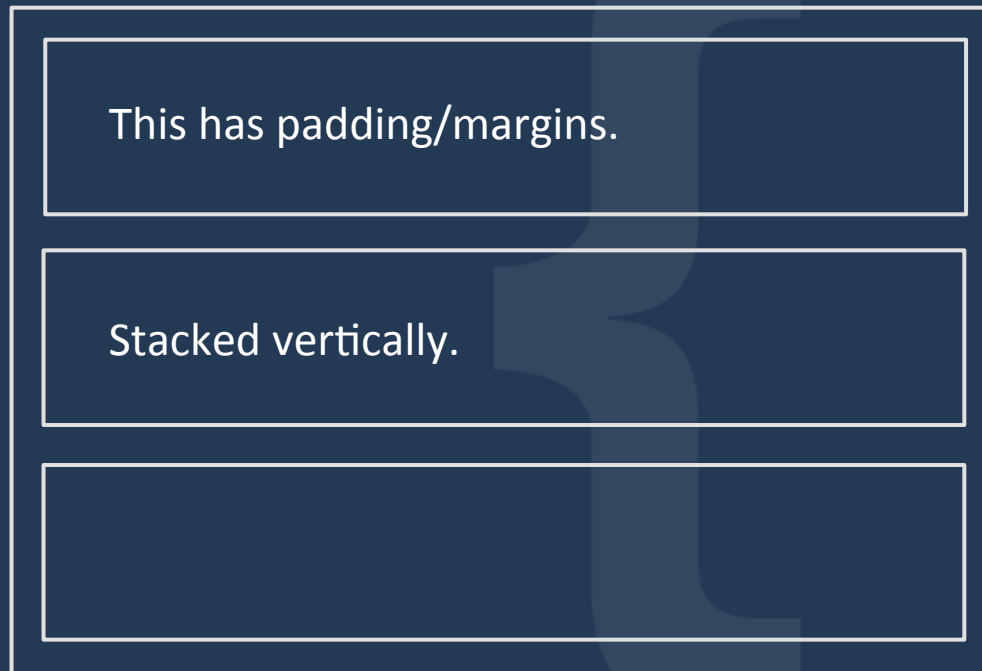
# CSS Box Model

- In order to stack horizontally, you must alter the CSS style for *display*
- **display**
  - There are 3 important values for display: *block*, *inline-block*, *inline*
- **block**
  - “block” displays respect all margins, paddings, height & width
  - Except, they have an auto-line break after it (next element goes directly below it)
- **inline-block**
  - Exactly like a block **but** it has no line break after it
- **inline**
  - Can have left & right margins/paddings, but *no top & bottom* ones. Also has no height/width. Allows elements next to them



# CSS Box Model

## Block Example

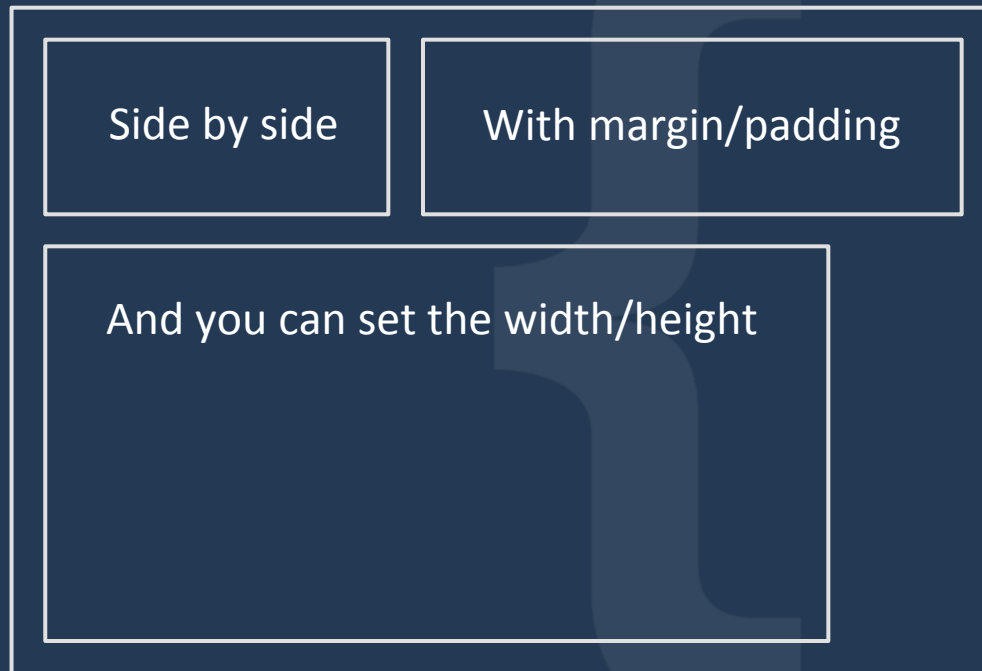






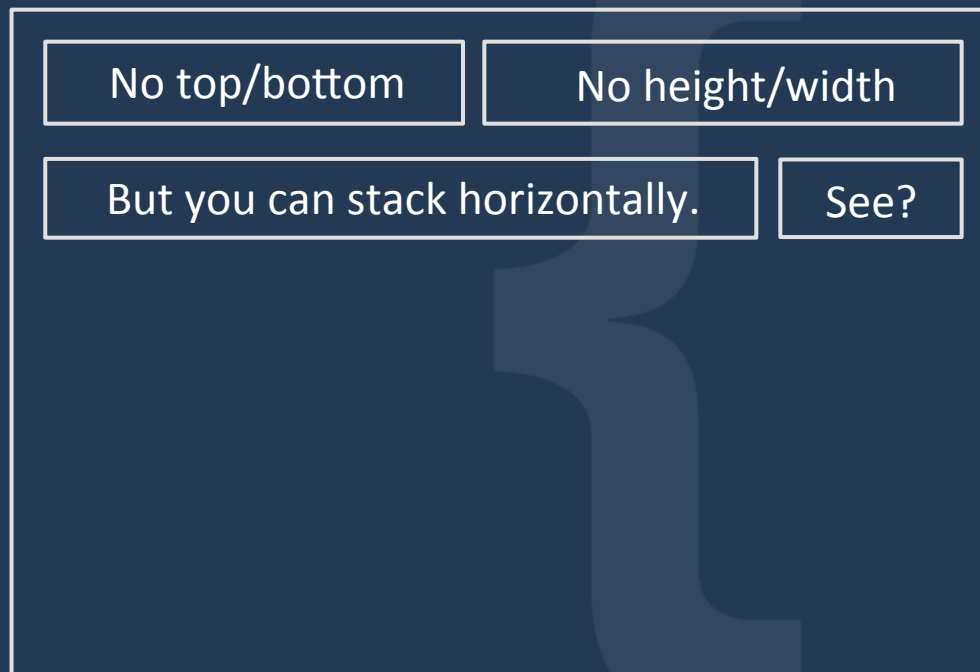
# CSS Box Model

## Inline-Block Example



# CSS Box Model

Inline Example (essentially a `<span>` tag)



# CSS Box Model - Examples

Inline Example (essentially a `<span>` tag)

The quick **brown fox** jumps over the lazy dog.

```
<div>
```

```
  The quick <span id="bold">brown fox</span> jumps over  
  the lazy dog.
```

```
</div>
```



# CSS Box Model - Examples

What happens if instead of `<span>`, you use `<div>`?

```
<div>
```

The quick `<div id="bold">`brown fox`</div>` jumps over the lazy dog.

```
</div>
```

The quick  
**brown fox**  
jumps over the lazy dog.

# CSS Box Model - Examples

Now if add the CSS properties to id="bold":  
*display: inline-block & padding: 0 40px*

```
<div>
```

The quick **<div id="bold">brown fox</div>** jumps over the  
lazy dog.

```
</div>
```

The quick

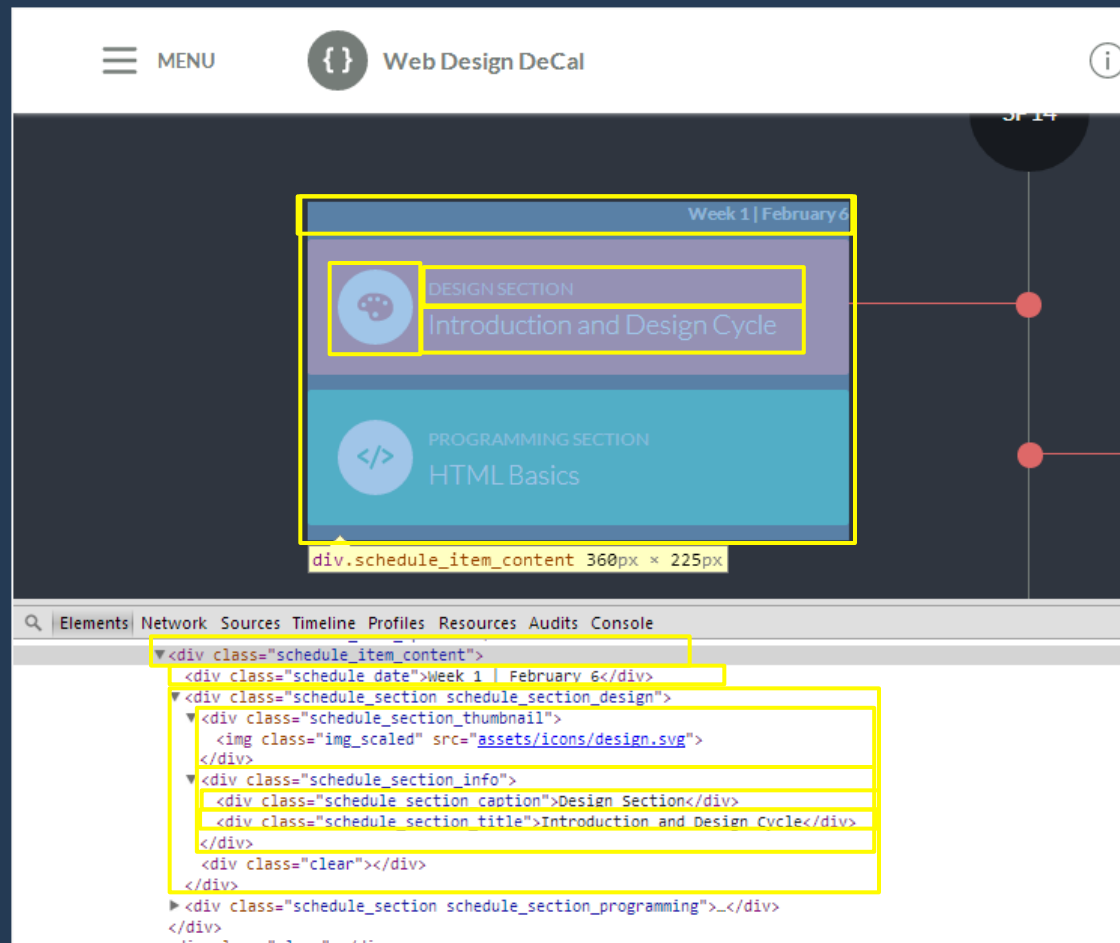
**brown fox**

jumps over the lazy dog.

You can add width, padding, and margins to your elements!  
Amazing!



# CSS Box Model

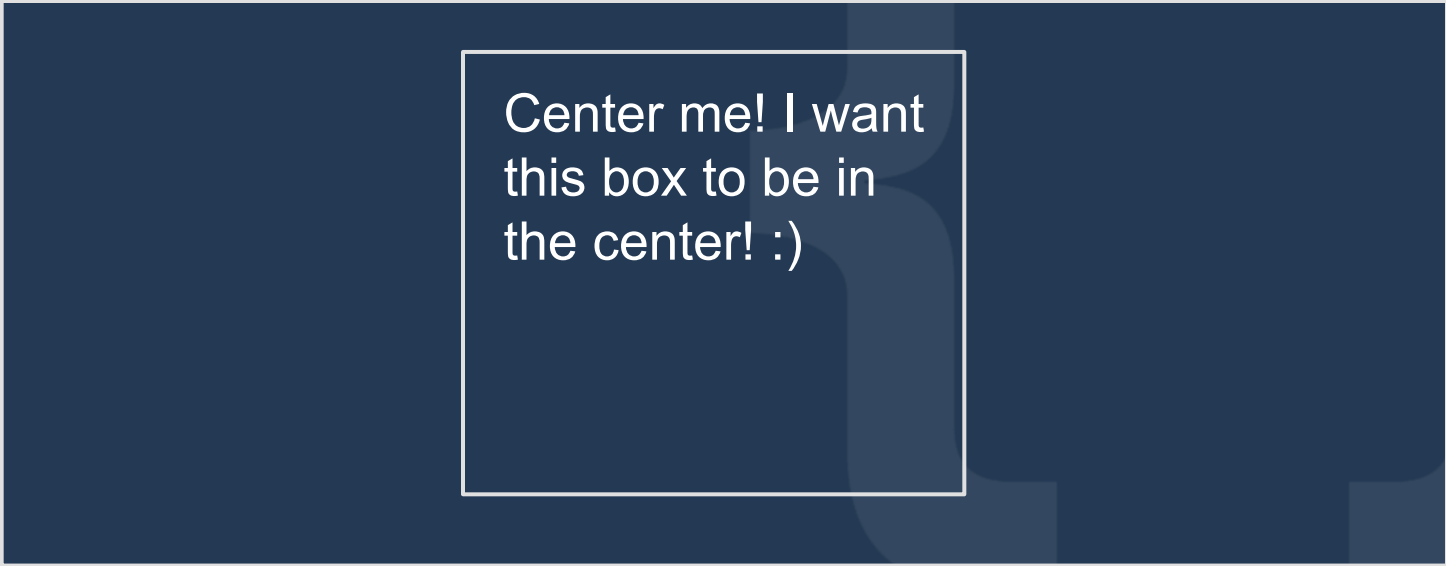




# *Demo*

# Centering a Div

- **Text or Div Element**
  - Centering text is easy, use *text-align: center*
  - What if you want to center a `<div>` that has a certain width?




Center me! I want  
this box to be in  
the center! :)



# Centering a Div

- **Text or Div Element**
  - By default, if you give your `<div>` a width (say 200px), it will automatically be on the left



Center me! I want  
this box to be in  
the center! :)

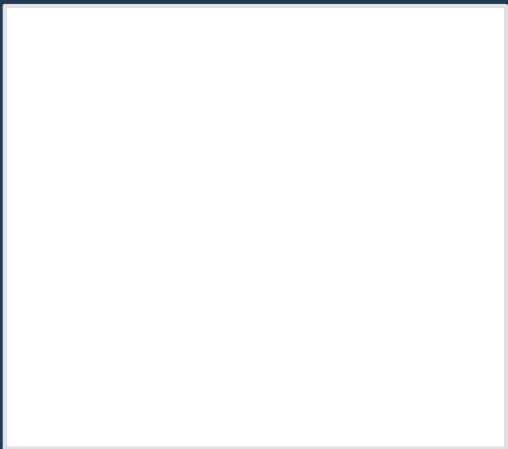
# Centering a Div

- **Text or Div Element**
  - To center a `<div>`, add 2 CSS properties to the element:
    - *`margin-left: auto; margin-right: auto;`*

Margin-left and margin-right auto lets the browser place your element in the center of page, or center of the above element (if it is not `<body>`)

# Centering an Image

- **Image**
  - What about centering an image? (`<img>` tags)
  - Again, by default it is on the left



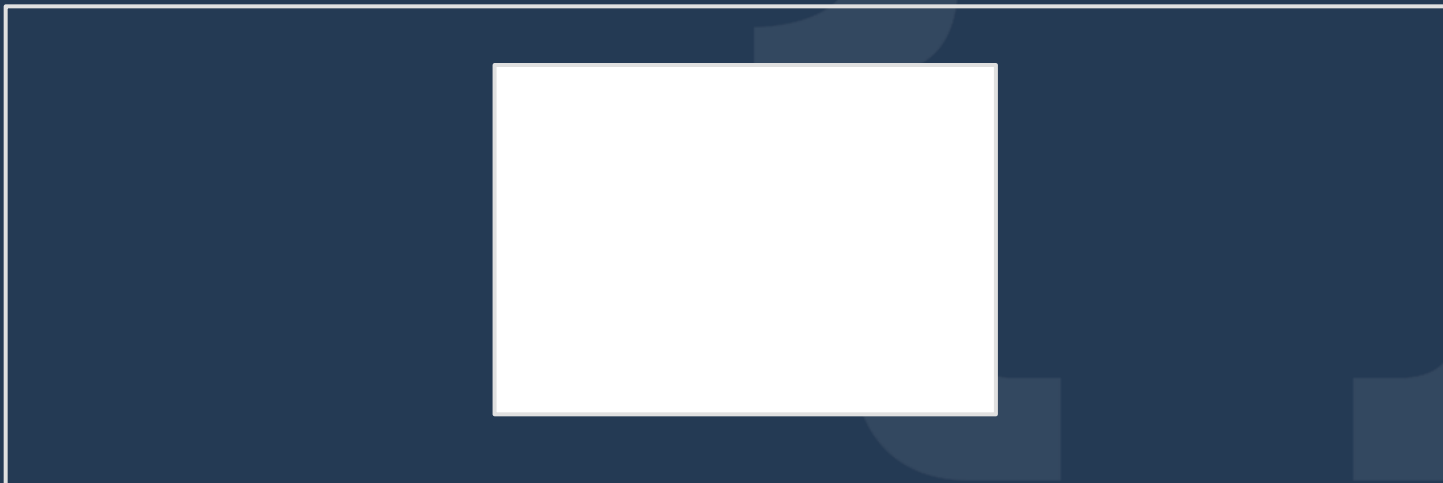
# Centering an Image

- **Image**
  - To center an image, like text, add *margin-left/right to auto*
  - Also, specifically add *display: block*



# Centering an Image

- **Image**
  - Remember that `<div>` elements by default are *display: block*
  - `<img>` tags by default are not *display: block*, so we must type in *display: block* manually, to tell the browser that the image takes up the entire row





# CSS Color



# CSS Color

- Color is important for font colors, background colors, borders, etc.!
- **Color** on the web can be represented in 3 ways:
  - A default color value
  - Hex Value
  - RGB Value
- What do these mean?
- **Default Color**
  - 16 pre-defined CSS colors
  - Example: *red, blue, black, white, maroon, etc.*
  - Too limited may not be what you want!

# CSS Color

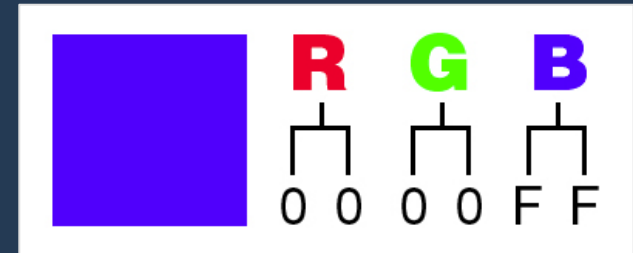
- **Hex Color** (Hexadecimal)
  - *Millions* of ways to define a color!
  - Syntax: Pound sign **#** followed by 6 digits/characters from 0 to 9 and A to F
  - Example: **#FF0000** (red), **#339CCD** (*light blue*), **#888888** (*gray*)

0-9 & A-F = 16 possible values  
 $16 * 16 * 16 * 16 * 16 * 16 = 16 \text{ Million Ways}$   
*To Define 1 Color!*





# CSS Color - Hex



- Hex values are 6 digits, or 3 bytes
- Each byte is 2 digits and represents a “color”
- **Red** corresponds to the 1<sup>st</sup> byte, **Green** to the 2<sup>nd</sup>, **Blue** to the 3<sup>rd</sup>
- A “0” indicates *no color*. Increasing the value to 1, 2, etc. increases the color. An “F” indicates *full color (lightest)*
  - Example: `#0000FF`
  - Equivalent to “no reds”, “no greens”, “full blues” = Pure Blue
- What is “Purple”?
- **Answer:** `#FF00FF`! It is a mix of pure red and pure blue. Other shades may include `#AA00AA` or `#330033`.

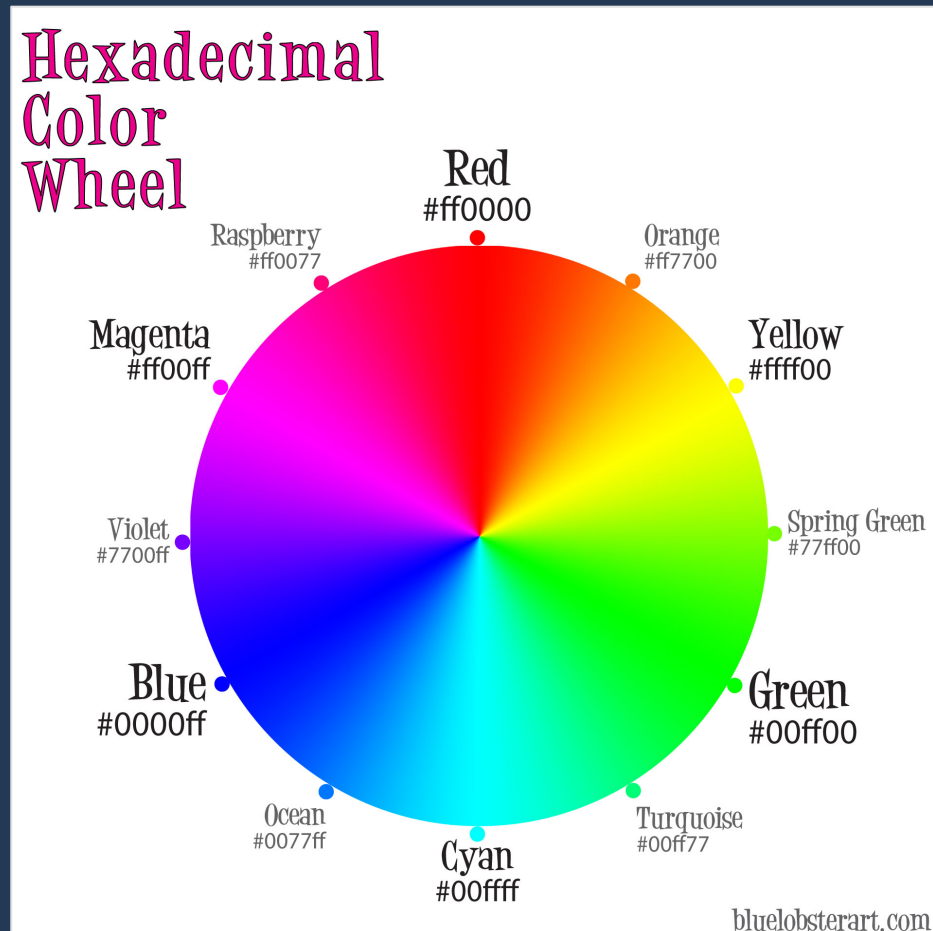


# CSS Color - Hex

- Screens and monitors are black by default
- When you have #000000 (no color), this means you get **black**
- When you have #FFFFFF (all colors), you get **white**
- Thus, some red on a black surface = dark red: #330000
- Another fun tip! If a color's byte has 2 repeating digits (88, FF, 00), and all 3 bytes have repeating digits, you can do a **shorthand** by using the single digit
  - Example: #FF0000 → #F00
  - #CC88DD → #C8D
  - #000000 → #000



# CSS Color - Hex



# CSS Color - RGB

- **RGB colors**
  - Syntax: *rgb(255, 0, 0)*
  - *rgb(...)* takes in 3 values: red, green, blue
  - Similar to Hex. 0 in rgb is 00 in hex, 255 in rgb is FF in hex
  - Examples:
    - *rgb(255, 0, 0)* vs. *#FF0000*
    - *rgb(51, 156, 205)* vs *#339CCD*
- Which to use? RGB or Hex?
  - Hex is great, supported by all browsers
  - Hex is used excessively in Photoshop, Illustrator, etc.
  - Easy to memorize over digits



# Summary

- Color is represented in 3 ways
  - Default colors
  - Hex colors
  - RGB colors
- The **CSS Box Model** has 4 components
  - Content
  - Padding
  - Border
  - Margin

All lecture material, handouts, and homework can be found at:  
<http://www.thewebdesignworkshop.co>



# Bonus Slides!

# CSS Color - RGBA

- **RGBA colors**
  - Like RGB, with another value: *Alpha*
  - Alpha controls the transparency of your color
    - *Ex:* 1 is the pure color, 0 is invisible, 0.6 is transparent
  - Syntax: `rgba(255,255,255,1)`
    - First three values are R, G, B from 0 to 255