

# HTML



# CSS



## HTML & CSS: LEVEL 1

Instructor: Sean Thompson

Week 3 - October 13, 2016

[seanmarshallthompson@gmail.com](mailto:seanmarshallthompson@gmail.com)



# SESSION OVERVIEW



- Review
- Anchor states and pseudo-elements
- Elements, Classes, and IDs
- HTML document organization overview: divs, sections, articles
- The CSS Box Model
- More CSS Styles and css abbreviations!



**REVIEW!**

# REVIEW: WEB GRAPHICS

## “WEB READY” GRAPHICS

-  Minimize file sizes to help load times in browser.
-  **Optimizes images for RGB displays** with correct **resolution** for browsers
- **Flattens** layers and removes metadata from graphics.

# REVIEW: WEB IMAGE TYPES

- **JPG or JPEG (Joint Photographic Experts Group)**  
is traditional for photos. Millions of colors, no transparency, no animation.
- **GIF (Graphics Interchange Format)** – 256 colors, can be animated, has transparency.
- **PNG 24 (Portable Network Graphic)** – Millions of colors, full alpha transparency, no animation.

# REVIEW: WEB IMAGE GOTCHAS

- Best practice to work in 72 PPI in graphic editor programs. (keeps file sizes down)
- Always work in **RGB** n working with graphics for the web. **CMYK** is for print.
- Make your designers stick to these rules!
- Graphics for **Retina devices** need to be saved out at 2X their “normal” size.

# <> REVIEW:BLOCK & INLINE ELEMENTS

## Block-level elements

- If no width is set, will expand naturally to fill its parent container
- If no height is set, will expand naturally to fit its child elements (assuming they are not floated or positioned)
- Can have margins and/or padding
- By default, will be placed below previous elements in the markup (assuming no floats or positioning on surrounding elements)
- Ex: **<div>**, **<h1>**, **<h2>**, **<p>**, **<ul>**, **<li>**

# <> BLOCK & INLINE ELEMENTS

## Inline elements

- Flows along with text content
- Will ignore top and bottom margin settings, but will apply left and right margins, and any padding
- Will ignore the width and height properties
- If floated left or right, will automatically become a block-level element, subject to all block characteristics
- Ex: **<a>**, **<em>**, **<strong>**, **<span>**



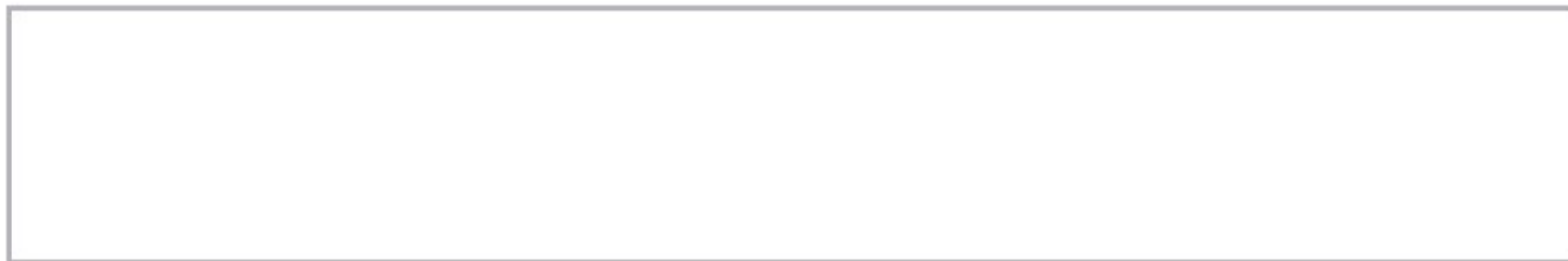
# <> BLOCK & INLINE ELEMENTS

## Block-level elements

**BLOCK ELEMENTS EXPAND NATURALLY** 



**AND NATURALLY DROP BELOW OTHER ELEMENTS** 



# <> BLOCK & INLINE ELEMENTS

## Inline elements

### INLINE ELEMENTS FLOW WITH TEXT

PELLENTESSQUE HABITANT MORBI TRISTIQUE SENECTUS  
ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS.  
VESTIBULUM **INLINE ELEMENT** VITAE, ULTRICIES  
EGET, TEMPOR SIT AMET, ANTE. DONEC EU LIBERO SIT  
AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI  
VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

# <> BLOCK & INLINE ELEMENTS

## “Inline-block” elements

- Inline/Block hybrid.
- Take up width and height like block-level elements.
- Flow with content around them.
- Can have margin and padding.
- Elements we know: **<img>** elements.

# REVIEW: <DIV> ELEMENTS

<div></div>

- <div> elements are generic **block elements**.
- Used to create **sections or groups** in HTML for layout.
- Can be used wrappers for other elements (including other divs!) for creating complex layouts.
- Have height and width
- **Building blocks of HTML layouts!**

# REVIEW: <SPAN> ELEMENTS

<span></span>

- <span> elements are generic **inline elements**.
- Can **nest inside** other block or inline elements.
- Used to **style unique inline content** or **content inside block elements**.
- Flow with content around them.

## <> REVIEW: MORE INLINE ELEMENTS

- **em** elements are used to show *emphasis*. (like italic).
- **strong** elements are used to show **importance in context** (like bold)

<p>"Oh, great. Someone ate <em>my only clean socks</em>." </p>

<p>"Was it <strong>the cat</strong>?"</p>

<p>"No, it was <strong>the dog</strong>."</p>

# REVIEW: CASCADING STYLE SHEETS

- Language for specifying how documents are presented to users
- We can override the browser's default presentation styles with our own.
- Provides consistent and scalable ways to **style single elements, single pages, or entire websites.**
- **Separates look and feel from content/markup.**

# REVIEW: ANATOMY OF A CSS RULE

```
selector {property: value;}
```

- **Selector** is the **thing** you want to style.
- **Property** is the **aspect/attribute** you want to style.
- **Value** is how you want to style it.
- **Values** always end in semicolons ( ; )

```
p {font-size: 14px; color: blue;}
```



# REVIEW: EXAMPLE CSS RULE

```
p {font-size: 14px;}
```

- **Selector** is the **p**. (<p> in the HTML)
- **Property** is the **font-size**.
- **Value** is **30px** (30 pixels high).
- All paragraph tags will have a font size of 14px.

# { REVIEW: MULTIPLE SELECTORS & PROPERTIES

- You can add multiple selectors to a CSS rule.
- You can add multiple properties to a CSS rule.
- Example: style all ordered and unordered lists:

```
ul,  
ol {  
    font-size: 16px;  
    font-weight: bold;  
    color: #444444;  
}
```

# REVIEW: CSS COMMENTS

```
<style>
  /* I am a CSS comment! */

  h1 { /* I am also a CSS comment */
    color: #ff0000;
  }
</style>
```

- Just like HTML, CSS can have **comments**.

# 🔗 REVIEW: COMMON FONT PROPERTIES

- **font-size:** a number followed by a measurement of how tall the element's text is, usually in ems (**em**) or pixels (**px**).
- **font-family:** the name of a typeface.
- **font-style:** (**normal**, or **italic** are most common)
- **font-weight:** **bold** (can also be values of 100, 200, up to 900 depending on the typeface).
- **line-height:** a number followed by a measurement of how tall the element's line of is, usually in ems (em) or pixels (px) (similar to **leading** in typography)

# 🔗 REVIEW: COLORS

- To set **text color**, the property is **color**.
- To set **background colors**, the property is **background-color**.
- Color **value** can be: **HEX**, **RGB**, or **RGBA**.
  - Hex: **#ffffff**
  - RGB: **rgb(245, 245, 245)**
  - RGBA: **rgba(245, 245, 245, 0.8)** – (0.8 represents alpha/opacity)

```
p {color: #222222;}
```

```
div {background-color: rgba(0,0,0,0.5);}
```

# {} REVIEW: WIDTH & HEIGHT

- Block elements have width and height by default, which you can override.
- You can set width and height of images with HTML attributes:

```

```

- **But** it's recommended to use CSS:

```
img { width: 300px; height: 200px; }
```

```
img { width: 300px; height: auto; }
```

# { CSS IN MULTIPLE PLACES

- **Inline styles** are applied to only a single element (best practice to avoid this if possible).
- **Internal styles** are added in the **<head>** of a page and style only that page.
- **External stylesheets** are called into multiple pages, and are declared in separate **.css** files. \*Best practice.

# { REVIEW: LINKING TO EXTERNAL STYLESHEET

```
<link href="css/styles.css" rel="stylesheet">
```

- Tells the browser to go find and load the CSS file.
- Goes inside the **<head>** element.
- Should go in every page that should load the styles.



# { REVIEW: THE “CASCADING” PART

The beauty of CSS is being able to create styles and then override them when you want to customize the look of your pages.

**There are three big rules for determining how styles get applied:**

- Styles are loaded from far to near.
- Styles are loaded from top to bottom.
- Children elements are more specific than parents.

# 🔗 REVIEW: STYLES “LOCATION”

Styles that are “closer” to the elements they style take precedence.

- Browser defaults
- External styles (in a **.css** file)
- Internal styles (in the **<head>**)
- Inline styles (on an element)

**Less  
Specific**



**Most  
Specific**

## 🔗 REVIEW: TOP TO BOTTOM

If the same property is styled multiple times for the same selector, **the last one sticks**.

```
p { color: #2f4251; }
```

```
ul{ color: #444444; }
```

```
/* some other stuff */
```

```
p { color: #daa645; } /* this one wins */
```

## 🔗 REVIEW: CHILDREN ARE SPECIFIC

Children elements **inherit** styles from their parents but can **override** parents with their own styles

```
body { color: #2f4251; } /* parent */
```

```
p { color: #daa645; } /* child */
```

## { REVIEW: SELECTORS CAN BE MORE SPECIFIC

If one style is **more specific** than another, it takes precedence

```
p { color: #daa645; } /* all paragraphs */
```

```
a { color: #e7c0c8; } /* links in general */
```

```
p a { color: #c4fe46; } /* a nested in p */
```

```
div p a { color: #a5dd5e; } /* a in p in div */
```

# QUESTIONS?



# ANCHOR STATES & CSS PSEUDO CLASSES

## ⌋ ANCHOR PSEUDO CLASSES

- **Pseudo-classes** are added to a selector to add **conditional styles** to an element.
- Most commonly used to style **states** of <a> and form elements.

```
a { /* default */ }
```

```
a:visited { /* a link that has been clicked */ }
```

```
a:hover { /* a link that has a mouse hover */ }
```

```
a:focus { /* a link that has keyboard focus */ }
```

```
a:active { /* a link that is being clicked */ }
```



# { :HOVER VS. :FOCUS

- **:hover** is for a link or other element that has a **mouse hover**.
- **:focus** is for a link or other element that has **keyboard focus**.

```
a:hover,  
a:focus {  
    /* often easiest to style them together */  
}
```

## {} OTHER PSEUDO CLASSES

- **:first-letter** styles the first letter of a block of text.
- **:first-child** and **:last-child** style the first and last children of a parent.
- **:nth-child()** can be used to style even or odd children, or to do math to style every 3rd or 5th, etc.
- **::selection** styles text that is selected by the user.



# ID & CLASS SELECTORS

# ELEMENT CSS SELECTOR

- Remember we can target all elements in CSS like so:

```
p {  
  /* all paragraphs on the page will be targeted */  
}
```

# CLASSES AND IDS

- **class** and **id** attributes can be added to any HTML element.
- **Classes** are for multiple elements on the page. (styles to re-used)
- **IDs** are for single, unique elements on a page.
- You can create whatever **class** and **id** values you want.

```
<div id="header"></div>
```

```
<div class="comment-box"></div>
```

# CLASS ATTRIBUTES

```
.comment-box {  
  width: 300px;  
  padding: 20px;  
  margin: auto;  
}
```

- **classes** can be shared by multiple elements on a page.
- Elements can have **multiple** classes.

```
<div class="comment-box bg-blue margin-sm"></div>
```

# CLASS SELECTORS IN CSS

- Start with a **period** (.)
- Can style any element with the class.

```
.kittens { width: 300px; }
```

- Or can be used to style only a **specific type** of element with the class.

```
h3.kittens { width: 400px; }
```

- Classes are **more specific** than an HTML selector.

# ID ATTRIBUTES

- **IDs** *cannot* be shared by multiple elements on a single page.
- Elements *cannot* have multiple IDs.

```
<div id="header"></div>
```

```
<div id="main"></div>
```

```
<div id="footer"></div>
```



# ID SELECTORS IN CSS

- Start with a **hash/pound sign (#)**
- Can style the single element with the ID.

```
#kittens { background-color: #000000; }
```

- IDs are **more specific** than class selectors!

# MIXING CLASS AND ID ATTRIBUTES

- Elements can have **id** and **class** attributes at the same time.

```
<div id="kittens">...</div>
```

```
<div id="puppies" class="small fluffy"></div>
```

```
<div id="birds" class="small feathery"></div>
```

- IDs selector styles can be used to override class selector styles.

# TIPS

- Recommended order of attack:
  - Type/element selectors
  - Class selectors
  - Descendant selectors
  - ID selectors
- If you overuse **IDs** in your styles, you're going to have a hard time.

# SEMANTIC ELEMENTS

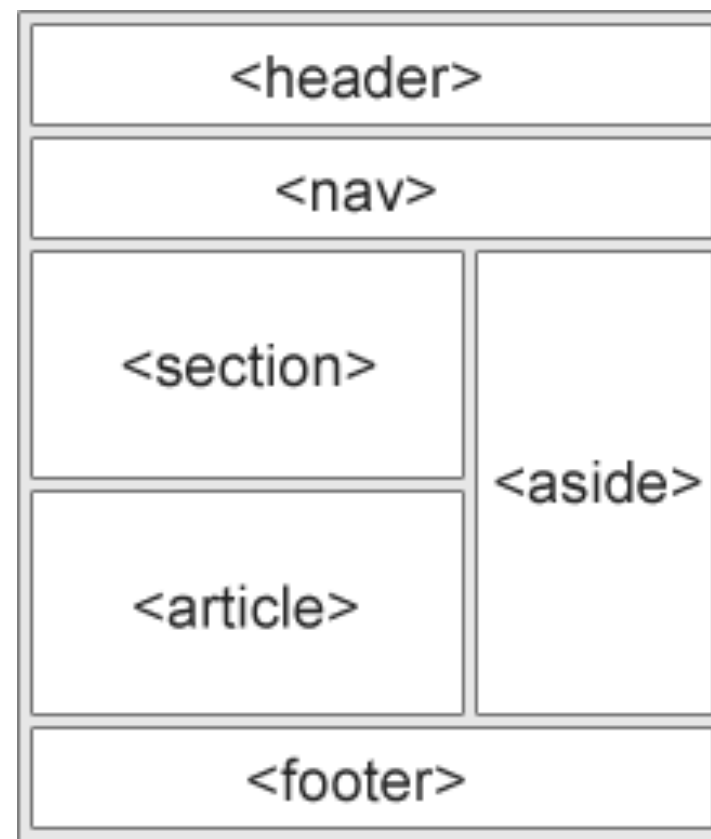
- **Semantics** is the study of the meanings of words and phrases in a language.
- **Semantic elements** = elements with a meaning.
- A semantic element clearly describes its meaning to both the **browser** and the **developer**.
- Non-semantic: `<div>`, `<span>`
- Semantic: `<form>`, `<article>`, `<section>`

# SEMANTIC ELEMENTS (HTML5)

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`
- `<figure>`
- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<section>`
- `<summary>`
- `<time>`

# FURTHER READING

- [http://www.w3schools.com/html/html5\\_semantic\\_elements.asp](http://www.w3schools.com/html/html5_semantic_elements.asp)





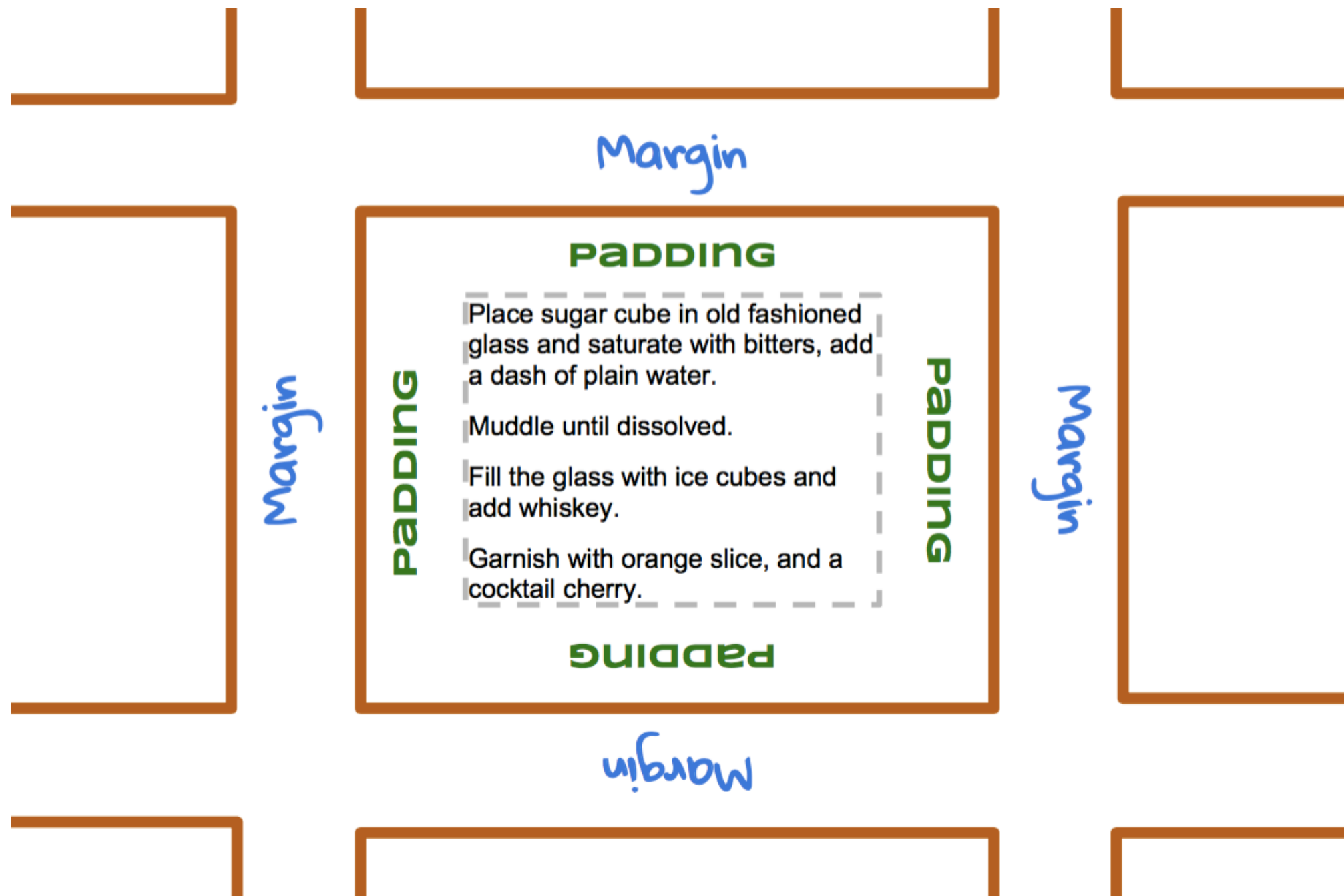
# THE CSS BOX MODEL

# CSS BOX MODEL

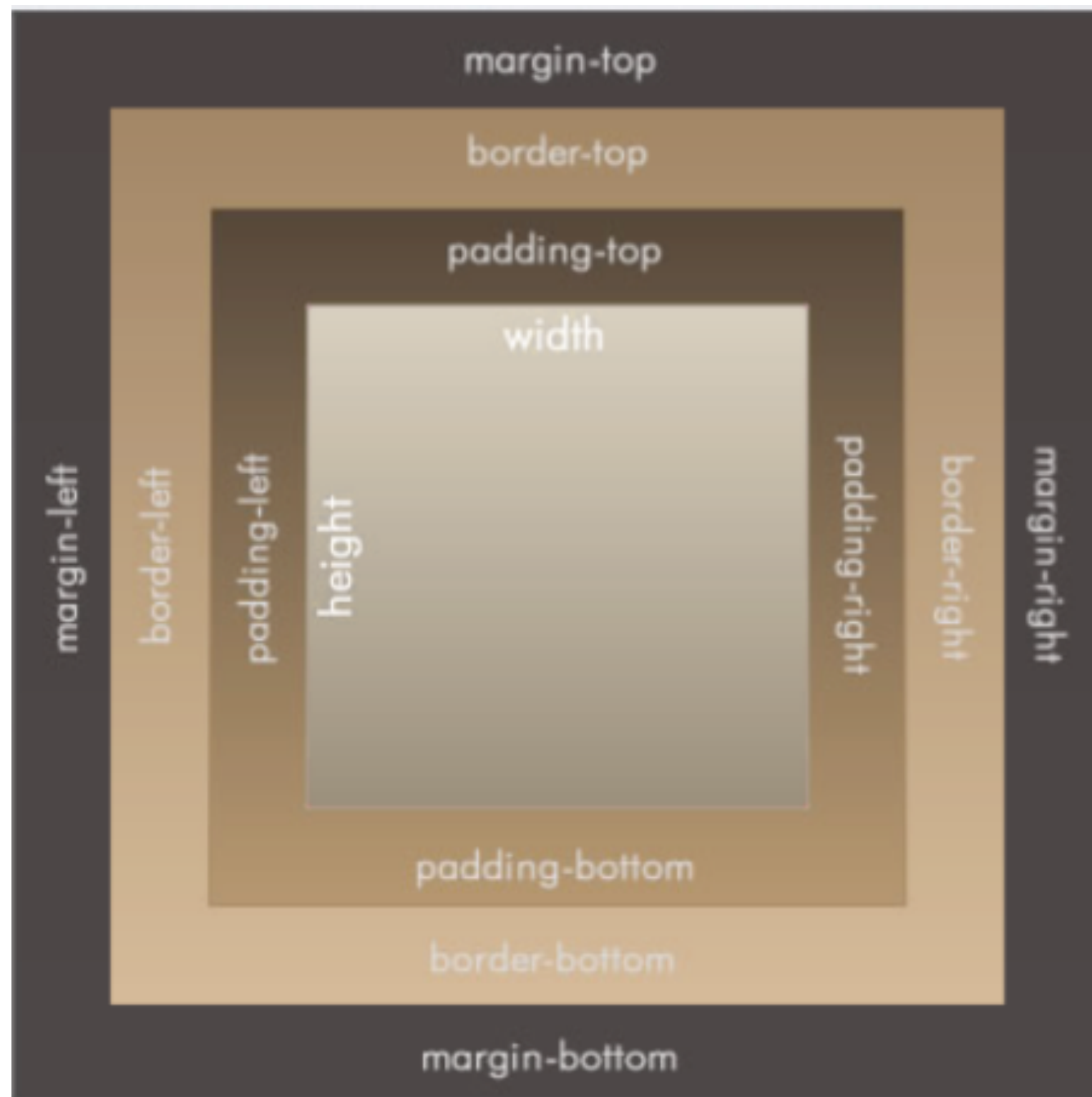
- **Content:** stuff in the box
- **Padding:** bubble wrap and packing peanuts
- **Border:** sides of the box
- **Margin:** space between multiple boxes
- In general, the box model applies to **block** and **inline-block** elements.



# CSS BOX MODEL



# CSS BOX MODEL



# BOX SIZING

```
body {  
    box-sizing: content-box; /* browser default */  
}
```

```
body {  
    box-sizing: border-box;  
}
```

- **border-box** includes border and padding **inside** of width (recommended)

# PADDING

- **padding** creates space between content and the **border** for readability/visual spacing.

```
#my-box {  
    padding-top: 20px;  
    padding-right: 40px;  
    padding-bottom: 40px;  
    padding-left: 20px;  
}  
/* or... */  
#my-box {  
    padding: 20px 40px 40px 20px;  
}
```

# PADDING

- If **top/bottom** and **left/right** padding match...

```
#my-box {  
    padding-top: 20px;  
    padding-right: 40px;  
    padding-bottom 20px;  
    padding-left: 40px;  
}  
/* COMBINE THEM! */  
#my-box {  
    padding: 20px 40px;  
}
```

# PADDING

- If **ALL** padding match...

```
#my-box {  
    padding-top: 20px;  
    padding-right: 20px;  
    padding-bottom 20px;  
    padding-left: 20px;  
}  
/* COMBINE THEM EVEN MORE! */  
#my-box {  
    padding: 20px;  
}
```

# MARGIN

- Goes outside the **border**.
- Creates space between the “boxes” of elements.
- Same abbreviation style as padding.
- Can take **negative** values to shift elements opposite direction.

```
#my-box {  
    margin-top: -20px;  
    margin-left: 30px;  
}
```

# MARGIN

- Goes outside the **border**.
- Creates space between the “boxes” of elements.
- Same abbreviation style as padding.
- Can take **negative** values to shift elements opposite direction.

```
#my-box {  
    margin-top: -20px;  
    margin-left: 30px;  
}
```

```
#my-box {  
    margin: 20px;  
}
```



# BORDER STYLES

- Allow you to specify the style, width, and color of an element **border**.

```
#my-box {  
    border-width: 4px;  
    border-color: #000000  
    border-style: dotted  
}
```

- Abbreviation:

```
#my-box { border 4px dotted #000000 }
```

# BORDER STYLES

- Learn more about border styles:
- <https://developer.mozilla.org/en-US/docs/Web/CSS/border>



# BACKGROUND STYLES

# BACKGROUND COLOR REVIEW

```
#my-box {  
  text-align: center;  
  color: #ffffff  
  background-color: rgba(0,0,0,0.5);  
}
```



# BACKGROUND IMAGES

- The property is **background-image**.
- The value is a **URL where the image lives**. (relative or absolute path)

```
#my-box {  
    background-image: url("images/kitten.jpg");  
}
```

# BACKGROUND IMAGES STYLES

- **background-repeat:** repeat/tile image horizontally or vertically; or not at all. (useful for patterns)
- **background-position:** Start at the left or right, top or bottom, center or not.
- **background-attachment:** Is it fixed or does it scroll with page?
- **background-size:** How much of the container does it cover?
- <https://developer.mozilla.org/en-US/docs/Web/CSS/background>

# BACKGROUND REPEAT

- **background-repeat: repeat-x;**  
/\* repeat the background horizontally \*/
- **background-repeat: repeat-y;**  
/\* repeat the background vertically \*/
- **background-repeat: no-repeat;**  
/\* don't repeat the background \*/

# BACKGROUND POSITION

- **background-position:** Values include both x-axis and y-axis.
- x-axis is first, y-axis is second.
- Can be **left/right top/bottom** or any measurement (px, %, ems, etc)

```
#my-box {  
    background-position: center center;  
}
```



# BACKGROUND POSITION EXAMPLES

`background-position: center center;`

`background-position: left top;`

`background-position: right bottom;`

`background-position: 10px 30px;`

# BACKGROUND ATTACHMENT

- **background-attachment: scroll**
  - background will scroll (default)
- **background-attachment: fixed**
  - background will stick fixed always

# “MAGIC” BACKGROUND IMAGE RECIPE

`/* make a fill-sized, fixed image background that covers the whole container */`

```
#header {  
    background-image: url("images/kittens.jpg");  
    background-repeat: no-repeat;  
    background-attachment: fixed;  
    background-size: cover;  
    background-position: center center;  
}
```



**PRACTICE TIME!**

# ASSIGNMENT

- Create a 1 page website that has a header, nav, main section, and footer, all in a main container div.
- Give them each unique IDs.
- Give them all a common class for base styles.
- Add two links to the nav.
- Add a background image in the header.
- Give your nav a background color and a border.
- Give your elements “breathing” room with padding and/or margin.
- Make use of some pseudo-class styles.