HTML & CSS: LEVEL 1

Instructor: Sean Thompson

Week 4 - December 1, 2016

seanmarshallthompson@gmail.com

SESSION OVERVIEW

- Review Box Model, Classes and Ids.
- Layout properties
- Floats and positioning
- Build a webpage with columns and floats

REVIEW!

{} 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.

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.

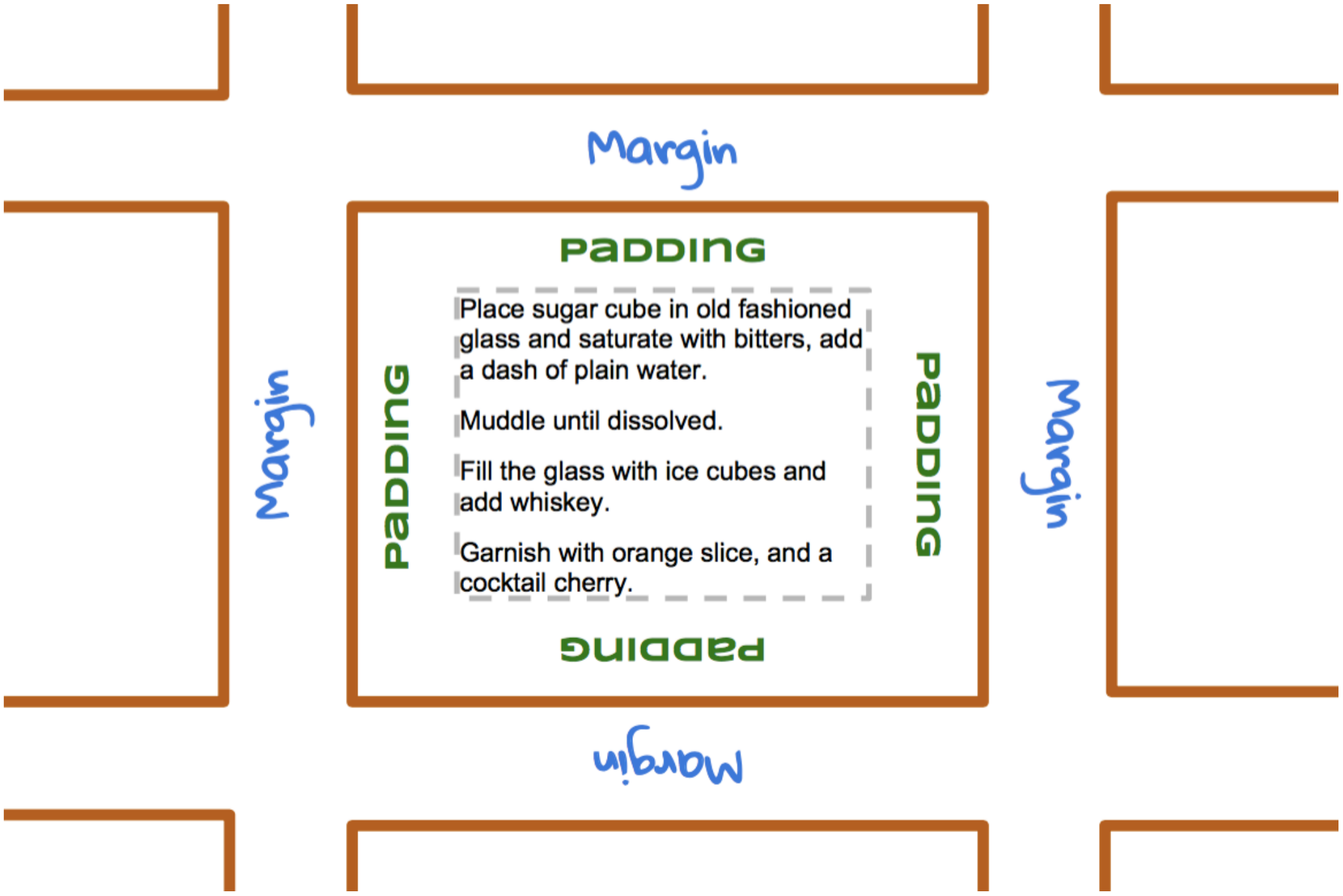
```
<div id="kittens">…</div>
<div id="puppies" class="small fluffy"></div>
<div id="birds" class="small feathery"></div>
```

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

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** ad **inline-block** elements.

CSS BOX MODEL



BOX SIZING

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

```
* {
    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;
}
```

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;
}
```

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}
```

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**

QUESTIONS?

WEB LAYOUTS

WEB LAYOUTS

- Before CSS, we used **<table>** elements to make layouts (bad!!)
- But now, with the advancements of CSS, we can use a variety of properties to arrange elements on the screen by adjusting the flow of the page.
- Basically, you can put elements anywhere.
  - (this can be a good and a bad thing!)

3 WEB LAYOUT PROPERTIES

- **display**: for dictating how elements behave within the box model. **(block, inline, inline-block)**
- **float:** for moving elements around within the page flow.
- **position:** for moving elements in and out of the page flow altogether.

THE DISPLAY PROPERTY

- Remember **block, inline, and inline-block** elements?
- You can tell elements to display differently using the CSS **display** property.
- Example:
  - display: block;
  - display: inline;
  - display: inline-block;

WHY USE DISPLAY?

- Make a **link** look more like a button.
- Add padding and margins to an inline element like a span.
- Make navigation links display horizontally.
- Make any text elements display inline.
- Make divs behave like images.

FLOATS!!!!!!

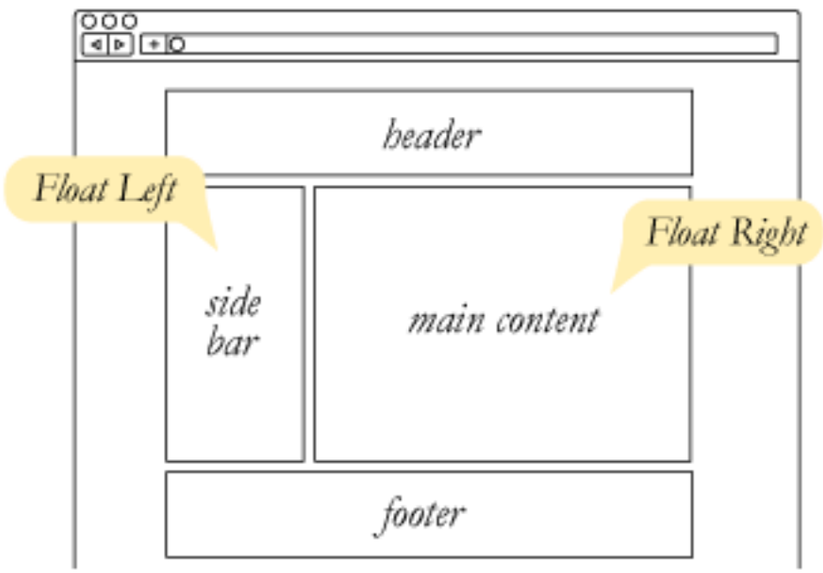
CSS FLOATS

- CSS **floats** can be tricky to grasp, but are foundational in creating complex web layouts.
- The **float** CSS property specifies that an element should be taken from the normal flow and **placed along the left or right side of its container**, where text and inline elements will wrap around it. (MDN)

CSS FLOATS

- Easiest way to offset content like divs, images, pullquotes, or other elements within the flow of a document.
- Requires that an element have **display: block;**
- **Three** possible values: **left, right, none;**
- **float: none;** is browser default.

```
#sidebar {
    float: left;
}
```



DEMO

THE CLEAR PROPERTY

- CSS float's sister is the **clear** property.
- An element that has the **clear** property set on it will not move up adjacent to the float like the float desires, but **will move itself down past the float.**

(uncleared float)

THE CLEAR PROPERTY

- One of the trickiest things about floats is when to stop "floating"
- You can give any element the **clear:both;** style to prevent it from floating.

```
#sidebar {
    clear: both;
}
```

THE MAGIC FLOAT FIX

- The most common fix today though is the self-clearing float.
- You can use a **pseudo-element** on the parent of the floated elements to create a “**self-clearing**” float.

```
.clearfix:after {
    content: “”;
    display: block;
    height: 0;
    clear: both;
}
```

DEMO

CSS POSITIONING

CSS POSITIONING

- The **position** property lets us arrange elements:
  - In relation to the normal flow (**relative)**
  - In a very specific place outside of the flow or within a **relative** element. (**absolute**)
  - In relation to the browser window (**fixed**)
- How **position** is applied depends on where the element is in the flow by default.

CSS POSITIONING

- We can dictate where elements go on the page down to the pixel!
- **left, right, top, bottom**
- Can tweak positively or negatively.

```
nav {
    position: absolute;
    right: -10px;
    top: 30px;
}
```

POSITION: FIXED

- **position: fixed;** is a way to make content "stick" to the browser window, regardless of where the user scrolls.
- Commonly used to make headers, nav, or footers that follow the page as it scrolls.

```
nav {
    position: fixed;
   width: 100%;
    left: 0;
    top: 0;
}
```

PRACTICE TIME!