I suggest reading the concepts from the original website. Those concepts will help you understand what we covered in class.

CSS Specificity

https://www.w3schools.com/css/css_specificity.asp

Box Model Fix

https://www.paulirish.com/2012/box-sizing-border-box-ftw/

CSS Layout - Float and Clear

https://www.w3schools.com/css/css_float.asp

CSS Layout - Position Property

https://www.w3schools.com/css/css_positioning.asp

CSS Gradients

https://www.w3schools.com/css/css3_gradients.asp

HTML responsive web design

https://www.w3schools.com/html/html_responsive.asp

CSS Transitions

https://www.w3schools.com/css/css3_transitions.asp

CSS Animations

https://www.w3schools.com/css/css3_animations.asp

If you do not have time to read the original website, read the following abstract.

CSS Specificity

https://www.w3schools.com/css/css_specificity.asp

If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out.

Think of specificity as a score/rank that determines which style declarations are ultimately applied to an element.

The universal selector (*) has low specificity, while ID selectors are highly specific!

Note: Specificity is a common reason why your CSS-rules don't apply to some elements, although you think they should.

How to Calculate Specificity?

Memorize how to calculate specificity!

Start at 0, add 1000 for style attribute, add 100 for each ID, add 10 for each attribute, class or pseudo-class, add 1 for each element name or pseudo-element.

Consider these three code fragments:

Example

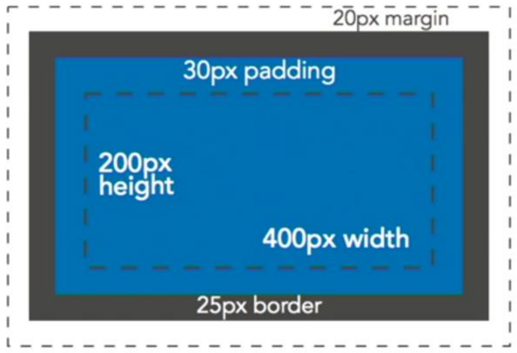
```
A: h1
B: #content h1
C: <div id="content"><h1 style="color: #ffffff">Heading</h1></div>
```

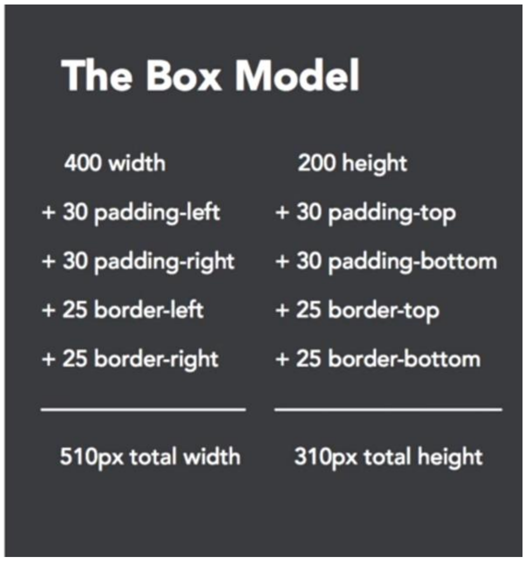
The specificity of A is 1 (one element)
The specificity of B is 101 (one ID reference and one element)
The specificity of C is 1000 (inline styling)

Since 1 < 101 < 1000, the third rule (C) has a greater level of specificity, and therefore will be applied.

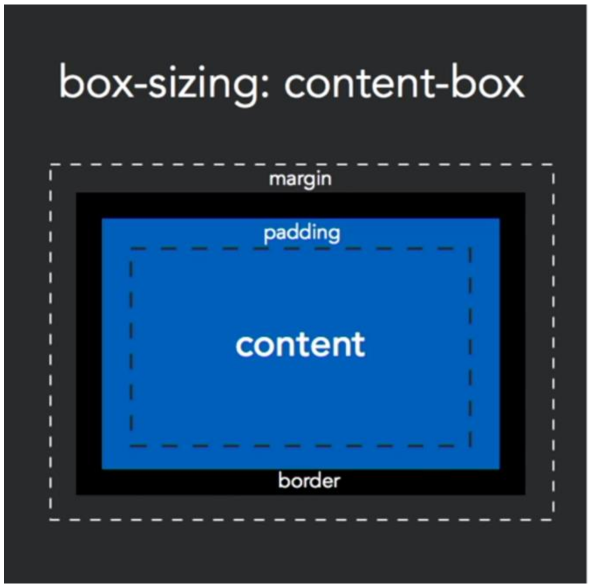
Box Model Fix

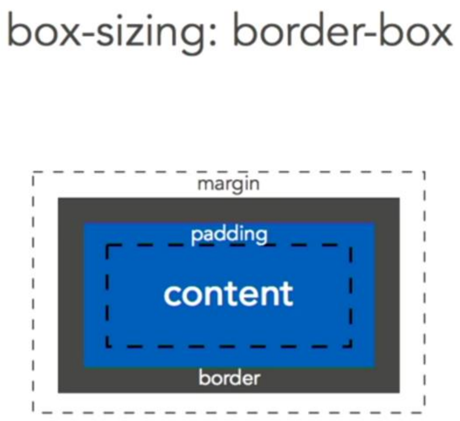
https://www.paulirish.com/2012/box-sizing-border-box-ftw/




The Box Model

400 width	200 height
+ 30 padding-left	+ 30 padding-top
+ 30 padding-right	+ 30 padding-bottom
+ 25 border-left	+ 25 border-top
+ 25 border-right	+ 25 border-bottom
510px total width	310px total height





One of my least favorite parts about layout with CSS is the relationship of width and padding. You're busy defining widths to match your grid or general column proportions, then down the line you start to add in text, which necessitates defining padding for those boxes. And 'lo and behold, you now are subtracting pixels from your original width so the box doesn't expand.

Ugh. If I say the width is 200px, gosh darn it, it's gonna be a 200px wide box even if I have 20px of padding. So as you know, this is NOT how the box model has worked for the past ten years.

Anyway, a recommendation for your CSS going forward:

```
/* apply a natural box layout model to all elements, but allowing components to change */
html {
  box-sizing: border-box;
}
*, *:before, *:after {
  box-sizing: inherit;
}
```

CSS Layout - Float and Clear

https://www.w3schools.com/css/css_float.asp

The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The `float` property can have one of the following values:

- left - The element floats to the left of its container
- right- The element floats to the right of its container
- none - The element does not float (will be displayed just where it occurs in the text). This is default
- inherit - The element inherits the float value of its parent

The clear Property

The `clear` property specifies what elements can float beside the cleared element and on which side.

The following example clears the float to the left. Means that no floating elements are allowed on the left side (of the div):

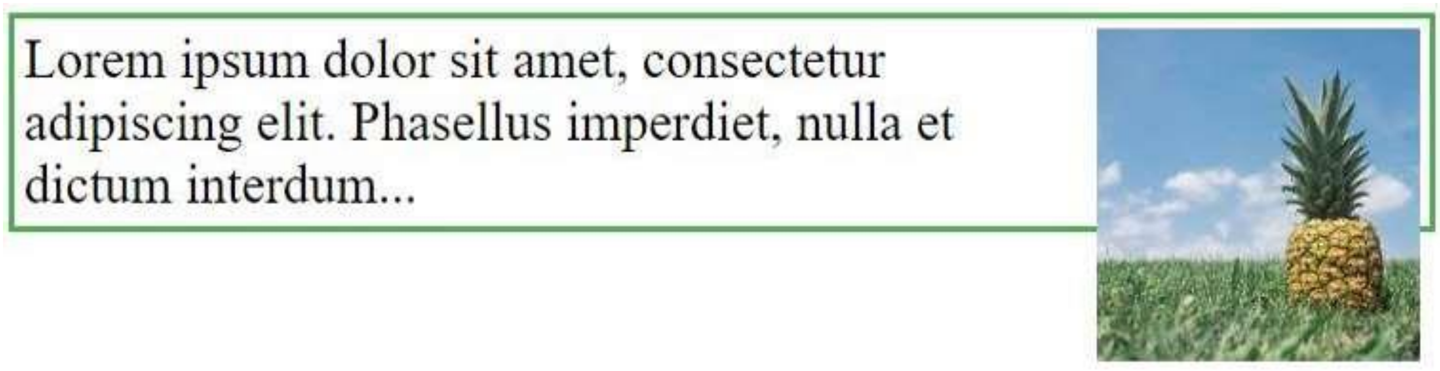
Example

```
div {
  clear: left;
}
```

The clearfix Hack

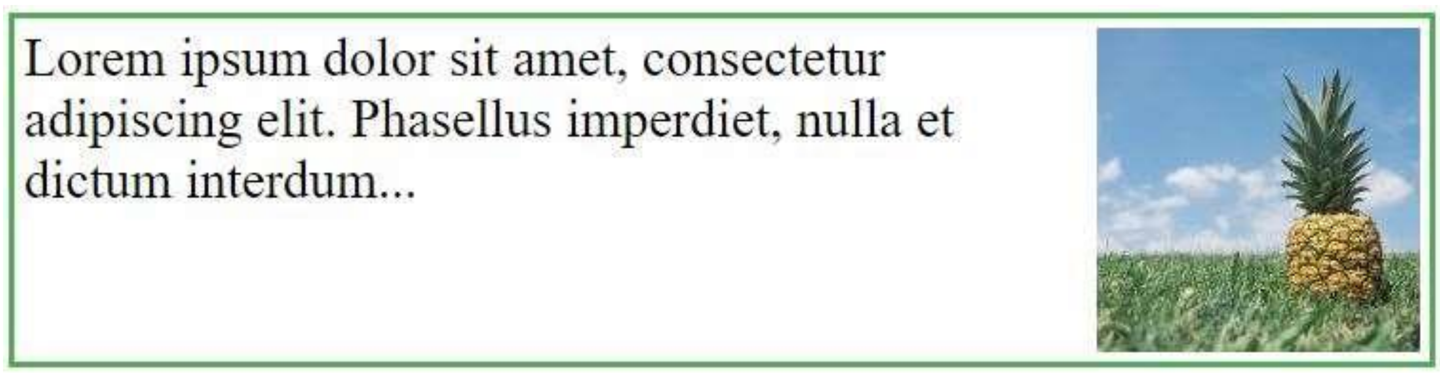
If an element is taller than the element containing it, and it is floated, it will "overflow" outside of its container:

Without Clearfix



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

With Clearfix



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

Then we can add `overflow: auto;` to the containing element to fix this problem:

Example

```
.clearfix {
  overflow: auto;
}
```

CSS Layout - Position Property

https://www.w3schools.com/css/css_positioning.asp

The `position` property specifies the type of positioning method used for an element.

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

CSS Gradients

https://www.w3schools.com/css/css3_gradients.asp

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

- **Linear Gradients (goes down/up/left/right/diagonally)**
- **Radial Gradients (defined by their center)**

Example

```
#grad {
  background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
}
```

CSS rgba() Function

Example

Define different RGB colors with opacity (RGBA):

```
#p1 {background-color:rgba(255,0,0,0.3);} /* red with opacity 0.3*/
```

0 means not transparent at all, and 1 means completely transparent.

HTML responsive web design

https://www.w3schools.com/html/html_responsive.asp

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones):

Responsive Images

Responsive images are images that scale nicely to fit any browser size.

Using the width Property

If the CSS `width` property is set to 100%, the image will be responsive and scale up and down:

Example

```
<img src="img_girl.jpg" style="width:100%;">
```

Using the max-width Property

If the `max-width` property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

Example

```
<img src="img_girl.jpg" style="max-width:100%;height:auto;">
```

Responsive Text Size

The text size can be set with a "vw" unit, which means the "viewport width".

That way the text size will follow the size of the browser window:

```
<h1 style="font-size:10vw">Hello World</h1>
```

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

Media Queries

In addition to resize text and images, it is also common to use media queries in responsive web pages.

With media queries you can define completely different styles for different browser sizes.

Example: resize the browser window to see that the three div elements below will display horizontally on large screens and stacked vertically on small screens:

```
.main {
  float: left;
  width: 60%; /* The width is 60%, by default */
}
/* Use a media query to add a breakpoint at 800px: */
@media screen and (max-width: 800px) {
  .left, .main, .right {
    width: 100%; /* The width is 100%, when the viewport is 800px or smaller */
  }
}
```

CSS Transitions

https://www.w3schools.com/css/css3_transitions.asp

CSS transitions allows you to change property values smoothly (from one value to another), over a given duration.

Example: Mouse over the element below to see a CSS transition effect:

To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

The following example shows a 100px * 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

Example

```
div {
  width: 100px;
  height: 100px;
  background: red;
  -webkit-transition: width 2s; /* Safari */
  transition: width 2s;
}
```

The transition effect will start when the specified CSS property (width) changes value.

Now, let us specify a new value for the width property when a user mouses over the <div> element:

Example

```
div:hover {
  width: 300px;
}
```

CSS Animations

https://www.w3schools.com/css/css3_animations.asp

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

When you specify CSS styles inside the `@keyframes` rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

Example

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```