Words

Enle Li + Liz Xiong

When applying a transition, you have a few decisions to make, each of which is set with a CSS property:

- Which CSS property to change (**transition-property**) (Required)
- How long it should take (**transition-duration**) (Required)
- The manner in which the transition accelerates (**transition-timing- function**)
- Whether there should be a pause before it starts (**transition-delay**)

Transitions require a **beginning state** and an **end state**. The element as it appears when it first loads is the beginning state. The end state needs to be triggered by a state change such as :hover, :focus, or :active…

CSS Animation Selectors

: hover

: focus

: active

transition-property

identifies the CSS property that is changing and that you want to transition smoothly. In our example, it's the background-color. You can also change the foreground color, borders, dimensions, font- and text-related attributes, and many more. TABLE 18-1 lists the animatab CSS properties as of this writing. The general rule is that if its value is a color, length, or number, that property can be a transition property.

Backgrounds

background-color
background-position

Borders and outlines

border-bottom-color
border-bottom-width
border-left-color
border-left-width
border-right-color
border-right-width
border-top-color
border-top-width
border-spacing
outline-color
outline-width

Color and opacity

color
opacity
visibility

Font and text

font-size
font-weight
letter-spacing
line-height
text-indent
text-shadow
word-spacing
vertical-align

Position

top
right
bottom
left
z-index
clip-path

Transforms

transform
transform-origin

Element box measurements

height
width
max-height
max-width
min-height
min-width
margin-bottom
margin-left
margin-top
padding-bottom
padding-left
padding-right
padding-top

Animateable CSS Properties

Timing Functions

```
. thisAwesomeClass {
        transition-timing-function :
                ease
                linear
                ease-in
                ease-out
                ease-in-out
                step-start
                step-end
                steps
                cubic-bezier(#,#,#,#)

}
```

the css property

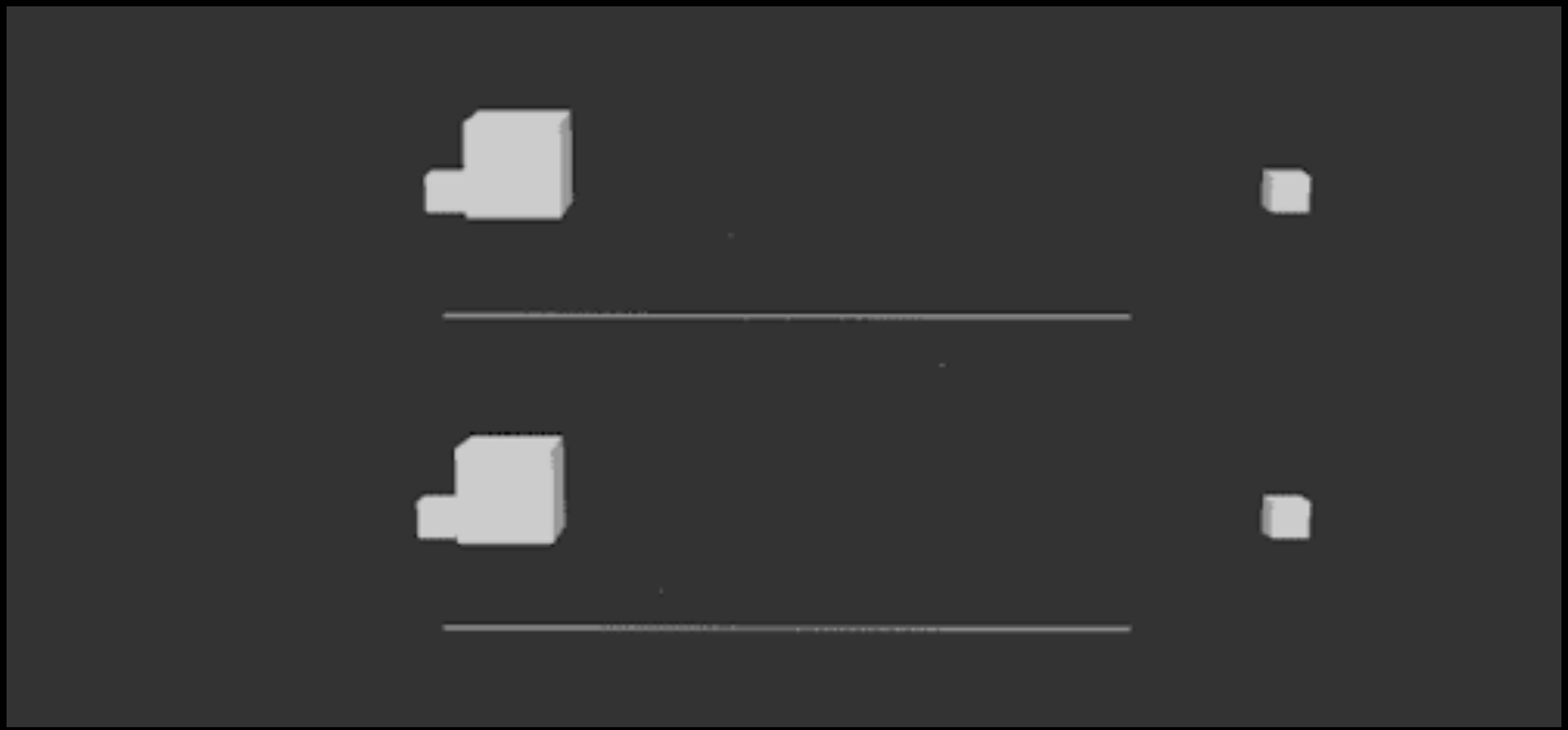
possible values you can set

The **property** and the **duration** are required and form the foundation of a transition, but you can refine it further. There are a number of ways a **transition** can roll out over time.

For example, it could start out fast and then slow down, start out slow and speed up, or stay the same speed all the way through, just to name a few possibilities. I think of it as the transition "style," but in the spec, it is known as the timing function or easing function.

The timing function you choose can have a big impact on the feel and believability of the animation, so if you plan on using transitions and CSS animations, it is a good idea to get familiar with the options.

Animation Principle #6 - Slow (Ease) In + Slow (Ease) Out



Animation Principle #9 - Timing

ease

Starts slowly, accelerates quickly, and then slows down at the end. This is the default value and works just fine for most short transitions.

linear

Stays consistent from the transition's beginning to end. Because it is so consistent, some say it has a mechanical feeling.

ease-in

Starts slowly, then speeds up.

ease-out

Starts out fast, then slows down.

ease-in-out

Starts slowly, speeds up, and then slows down again at the very end. It is similar to ease, but with less pronounced acceleration in the middle.

cubic-bezier(x1,y1,x2,y2)

The acceleration of a transition can be plotted with a curve called a Bezier curve. The steep parts of the curve indicate a fast rate of change, and the flat parts indicate a slow rate of change.

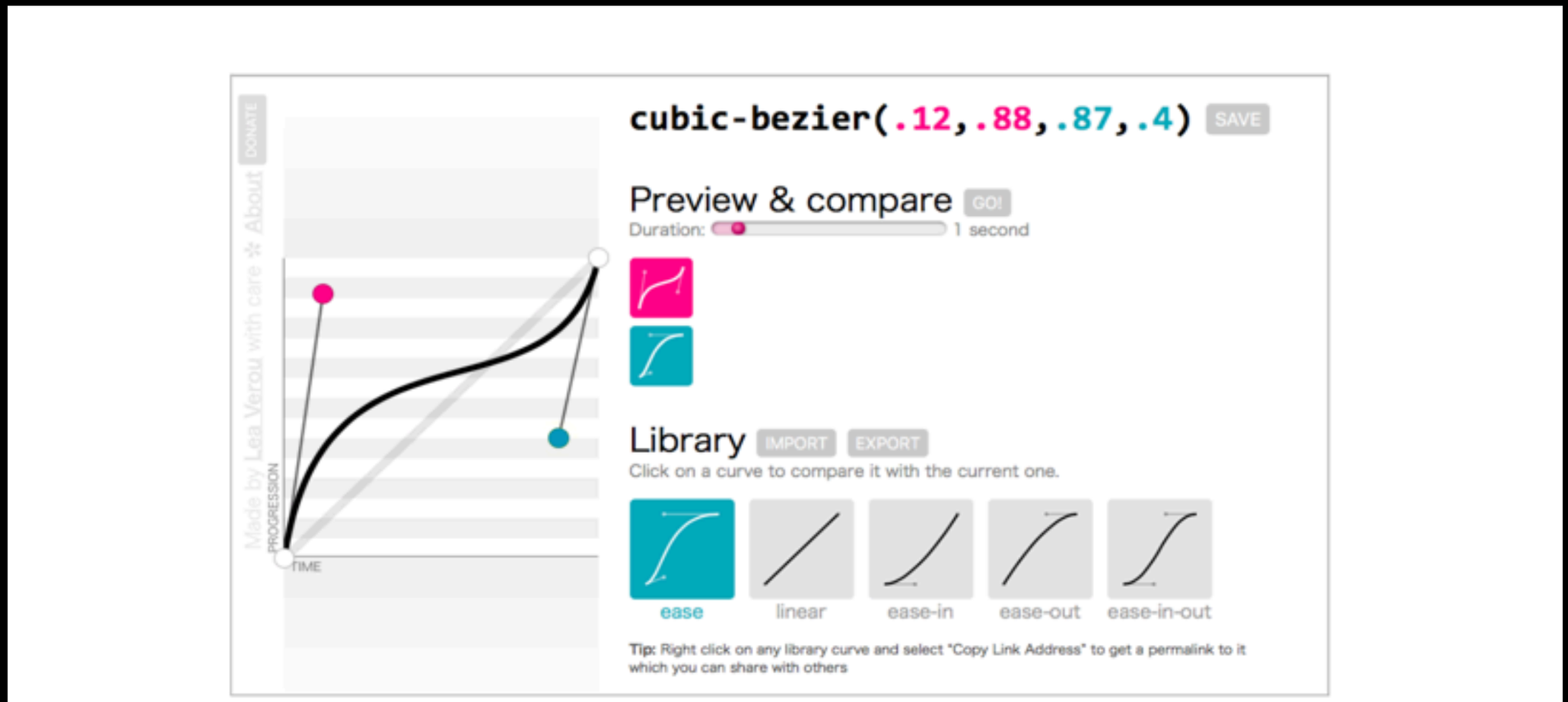


FIGURE 18-2. Examples of Bezier curves from Cubic-Bezier.com. On the left is my custom curve that starts fast, slows down, and ends fast.

You can see that the ease curve is a tiny bit flat in the beginning, gets very steep (fast), then ends flat (slow). The linear keyword, on the other hand, moves at a consistent rate for the whole transition.

You can get the feel of your animation just right by creating a custom curve. The site Cubic-Bezier.com is a great tool for playing around with transition timing and generating the resulting code. The four numbers in the value represent the x and y positions of the start and end Bezier curve handles (the pink and blue dots.

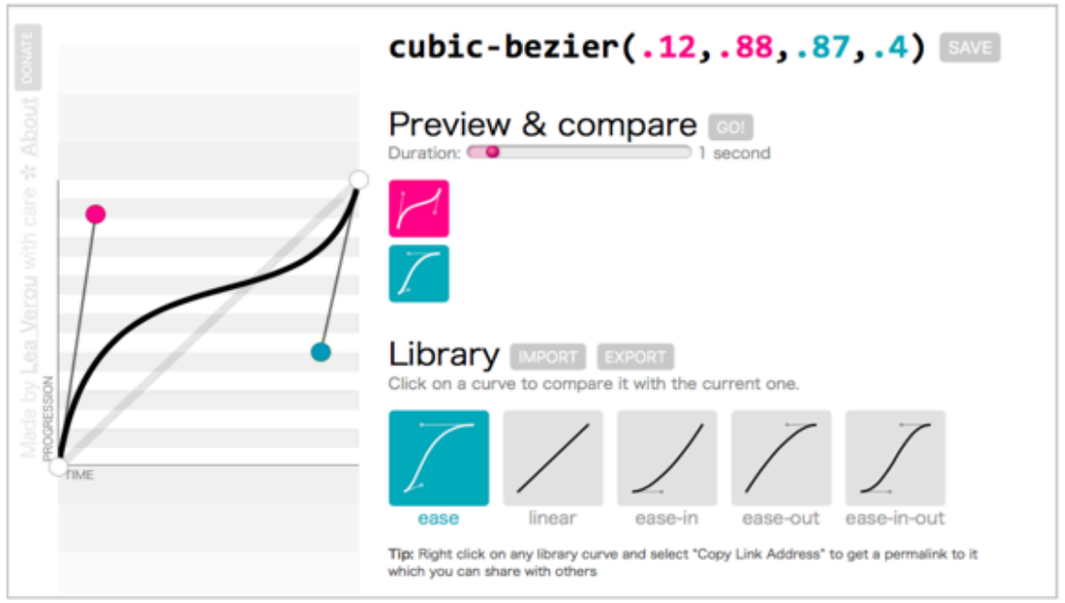


FIGURE 18-2. Examples of Bezier curves from Cubic-Bezier.com. On the left is my custom curve that starts fast, slows down, and ends fast.

steps(#, start|end)

Divides the transitions into a number of steps as defined by a stepping function. The first is the number of steps, and the **start** and **end** keywords define whether the change in sta happens at the begin- ning (**start**) or end of each step. Step animation is especially useful keyframe animation with sprite images. For a better explanation and examples, I recomm the article “Using Multi-Step Animations and Transitions,” by Geoff Graham on CSS-Tricks (*css-tricks.com/using-multi- step-animations-transitions/*).

step-start

Changes states in one step, at the beginning of the duration time (the same as **steps(1,st** The result is a sudden state change, the same as if no transition had been applied at all.

step-end

Changes states in one step, at the end of the duration time (the same as **steps(1,end)**).

transition-delay

The transition-delay property, as you might guess, delays the start of the animation by a specified amount of time.

The Shorthand transition Property

The authors of the CSS3 spec had the good sense to give us the shorthand transition property to combine all of these properties into one declaration.You've seen this sort of thing with the shorthand border property. Here is the syntax:

```
transition: property duration timing-function delay;
```

```
.theClass {
    transition: background-color 0.3s ease-in-out 0.2s;
}
```

The values for each of the **transition-*** properties are listed out, separated by character spaces. The order isn't important as long as the **duration** (which is required) appears before **delay** (which is optional). If you provide only one time value, it will be assumed to be the duration.

The sub-properties of the **animation** property are:

animation-delay	Configures the delay between the time the element is loaded and the beginning of the animation sequence.
animation-direction	Configures whether or not the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself.
animation-duration	Configures the length of time that an animation should take to complete one cycle.
animation-iteration-count	Configures the number of times the animation should repeat; you can specify infinite to repeat the animation indefinitely.
animation-name	Specifies the name of the **@keyframes** at-rule describing the animation's keyframes.

animation-play-state	Lets you pause and resume the animation sequence.
animation-timing-function	Configures the timing of the animation; that is, how the animation transitions through keyframes, by establishing acceleration curves.
animation-fill-mode	Configures what values are applied by the animation before and after it is executing.

@keyframes + animation property

misc css stuff

calc

you can use the calc CSS function to define numeric values in terms of expressions:

/* property: calc(expression) */

```
div {

  width: calc(50% - 10px);
  width: calc(100% / 6);

}
```

CSS variables

variables are a relatively new CSS feature.

CSS variables are entities defined by CSS authors that contain specific values to be reused throughout a document. They are set using custom property notation (e.g., --main-color: black;) and are accessed using the **var( )** function (e.g., color: var(—main-color);).

CSS variables are subject to the cascade and inherit their value from their parent.

```
:root {
  --thisGreat-color: black: hotpink;
}

h1 {
  background-color: var(--thisGreat-color);
}
```

background-properties

An easy way to render images stretched and cropped to a given size: set it as a background image for an element.

```
header {
  background-image: url(globe.png);
  height: 300px;
}
```

the world is big

background-properties

```
background-size: cover;
background-size: contain;
background-repeat: no-repeat;
background-position: top;
background-position: center;
```

Midterm Schedule + Details

Midterm Accounts for 20% of Final Grade

October 22 + 24: User Testing
October 29 @ 11:59pm: Final Project pushed to yr Github page repo + Linked posted on the Wiki (url forthcoming)

***** Participation in User Testing is MANDATORY**
+ Counts toward your Class Participation Grade

This means both as a tester, a user, a presenter + audience member.

If you miss a day of user testing or oral presentations you **MUST** have a valid reason + email me well in advance. Otherwise both your project grade + class grade will be affected.

This also means that while your classmates are presenting your laptop lids are CLOSED. If you want to take notes you can do so with pen + paper.

Midterm Grade Breakdown

	EXCEEDED (99)	MET CRITERIA (90)	APPROACHED (80)	DID'T MEET CRITERIA (60)	SCORE
Project concept	Something totally original or new.	Interesting idea or a great take on standards.	Nothing that new but you added nice personal touches.	Little thought or effort put into your concept.	**10**
Design Strategy	Overall aesthetic + UI are exceptional	Overall aesthetic is cohesive, interesting + UI is intuitive.	Looks good, but not overall aesthetically cohesive. Confusing interaction.	Little thought went into the aesthetic or Interaction design.	**15**
Technical Execution	Exceeds requirements.	Meets all requirements.	Most requirements met.	Did not meet requirements**.**	**40**
Participation	Close attention to your user feedback, enthusiastic + generous feedback as a user + fully attentive to others' work.	Attention to user feedback, feedback as a user + attentive to others' work.	Little attention to classmate's work as presenters, users or testers.	Disrespectful lack of attention to classmate's work as presenters, users or testers.	**15**
Presentation	Engaging oral + visual presentation with insight into your process, ample technical details, user feedback + how you responded.	Good presentation with some insight into your process, technical details, user feedback + how you responded.	Completed presentation with little insight into your process, technical details, user feedback or how you responded.	Unclear presentation with little context as to how or why you made this project.	**10**
On time, commented code w/ sources cited	On time, fully functional, with extensive comments.	On time, fully functional, with comments.	On time, functional, with little comments.	Late, broken code +/or no comments.	**10**