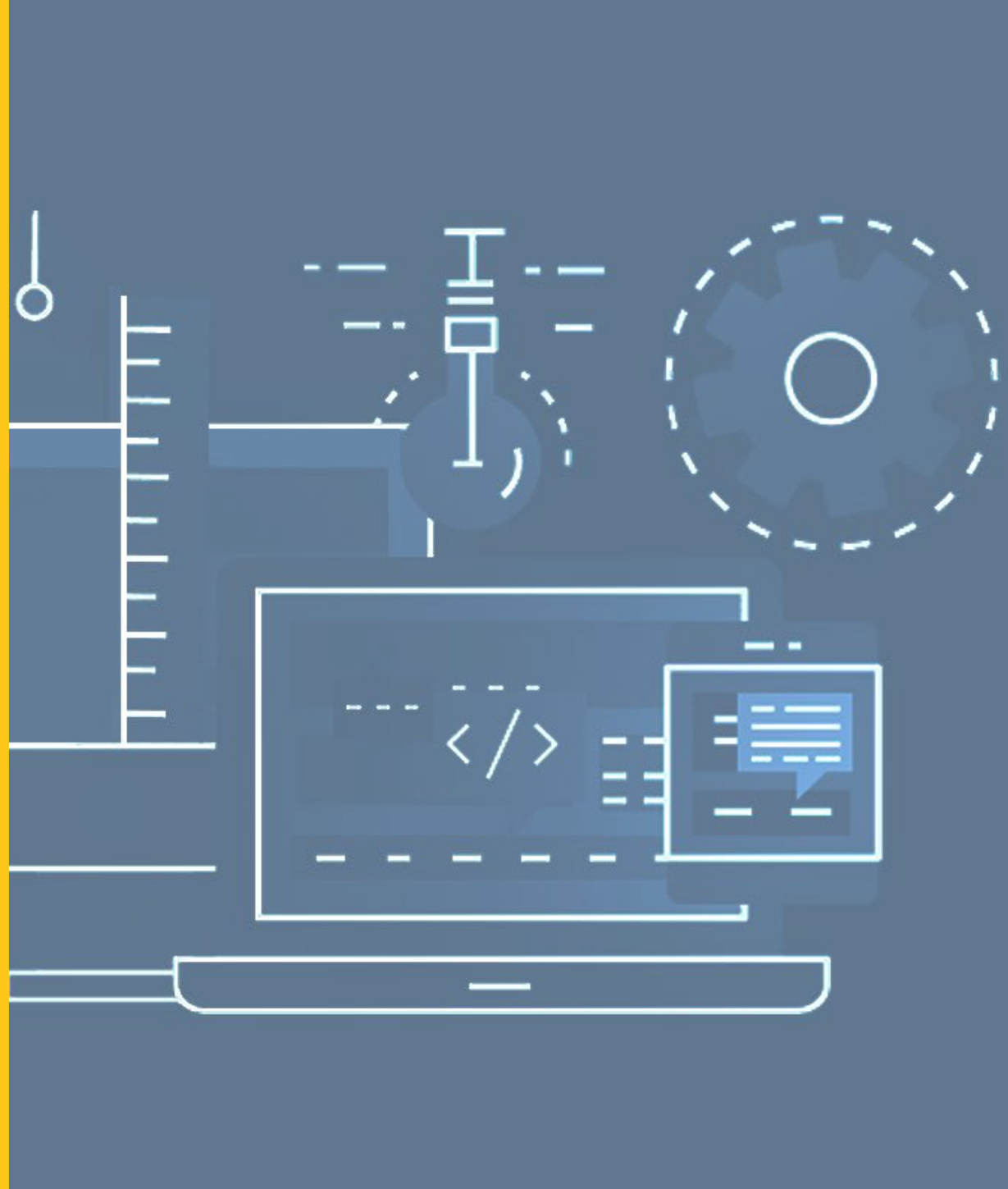
SHELLY GRAHAM, 02/03/2022

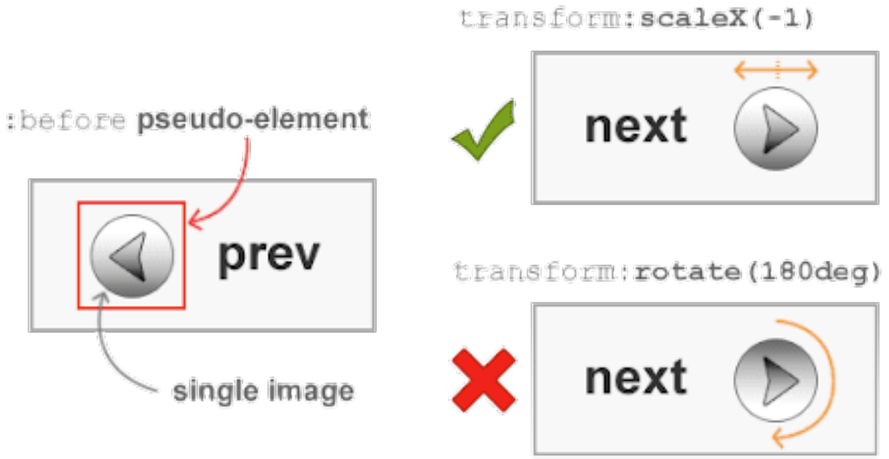
WEB DEV 3 WINTER 2022

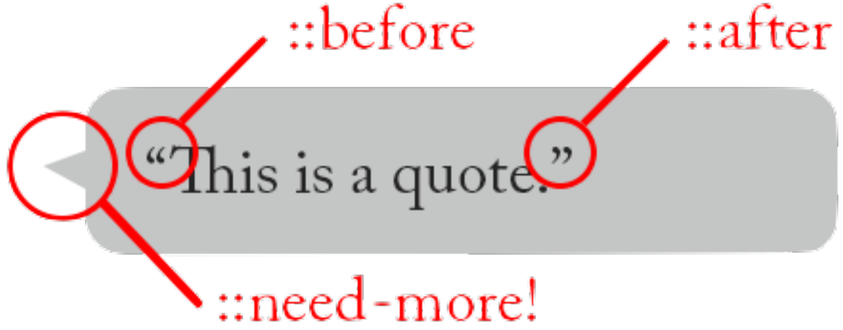
Week 4:
Transitions, Backgrounds, SVGs
PART 2



PSEUDO ELEMENTS

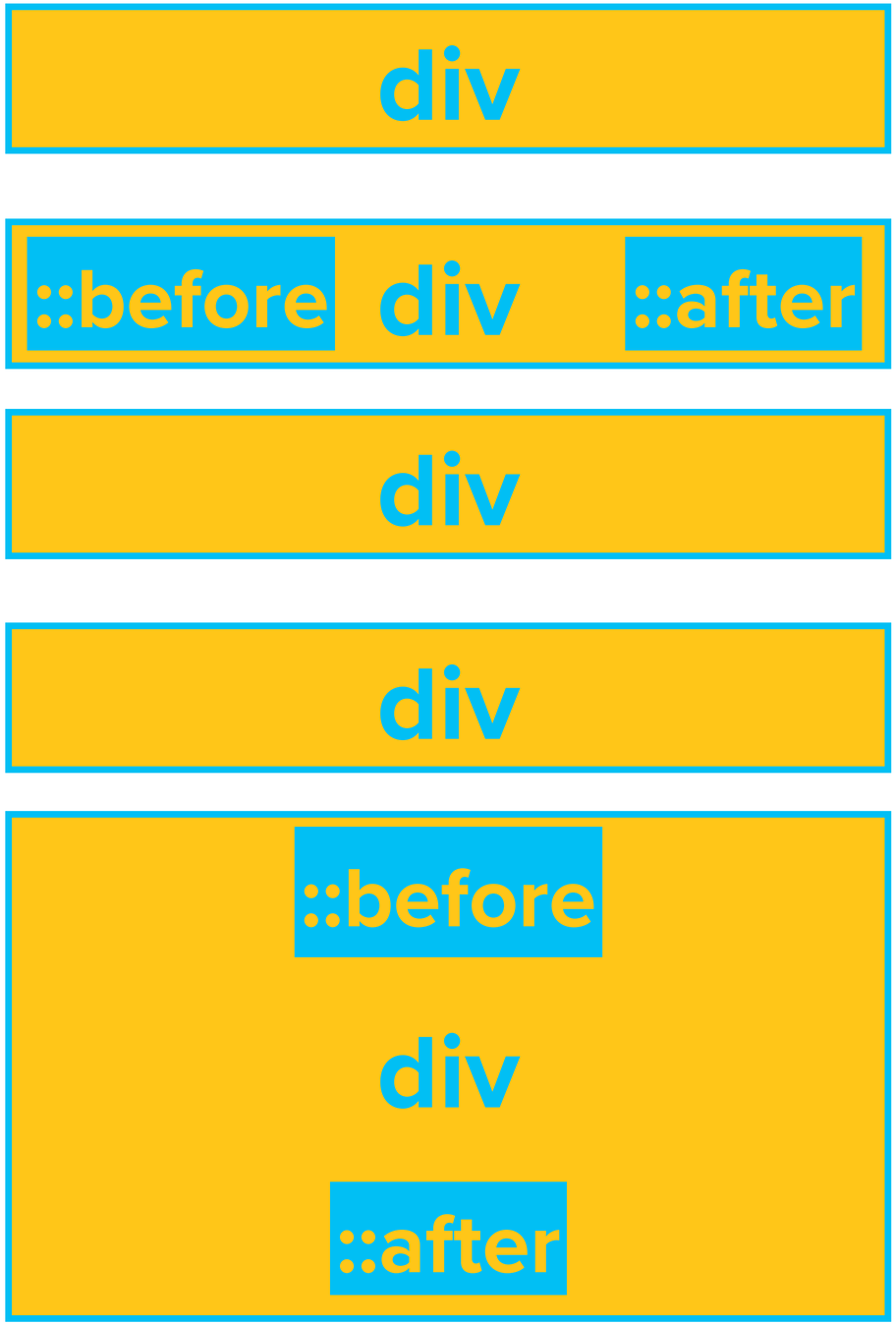
- Pseudo elements are keywords you can add to any selector, that allows you to style a specific part of said selector
- Great way to add extra styling without adding extra markup in your HTML
- There are several pseudo elements but the most important ones are ::before and ::after
- Syntax: double colon ::before, ::after
- Needs the “content” attribute to work. This can be in single or double quotes
- —> Content tells pseudo element: there is some content, even if we set it to an empty string
- Pseudo elements will show via Dev Tools in Inspector





PSEUDO ELEMENTS

- Pseudo elements don't get inserted before the element you bind it to but before it's content!
- It's include within the element you you bind it to, not before or after but at the beginning or end of it
- You can style them completely separate from the content element it's attached to
- I recommend to add your pseudo elements to universal selector and:
  box-sizing: border-box
- Pseudo elements don't work on images!
- If you want to add text, simply type in content string



PSEUDO ELEMENTS AND PSEUDO CLASSES

- Pseudo classes are keywords added to a selector that specifies a special state of the selected element
- If you want to use a pseudo class on a pseudo element, the class has to come BEFORE the element:

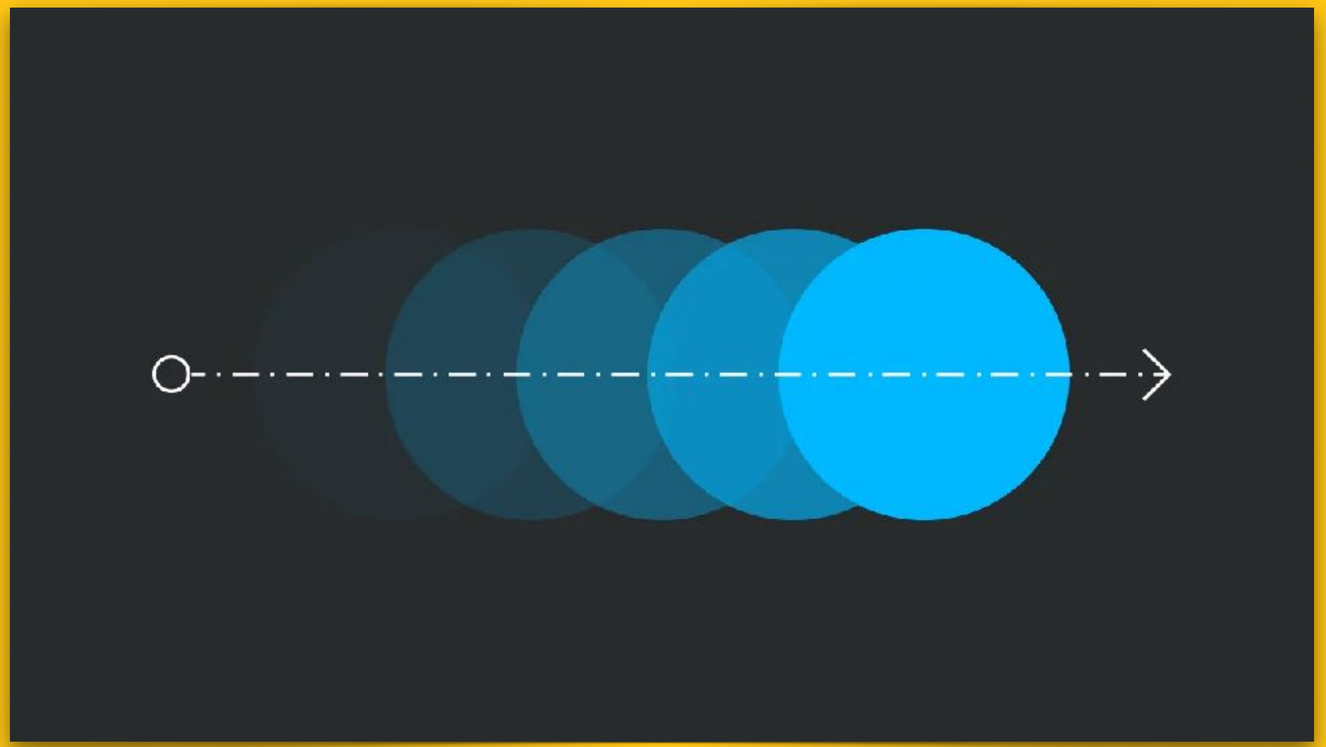
```
p:hover::before {
    background: red;
}
```

- Pseudo classes —> single colon:
  - :hover
  - :focus
  - :active
- Pseudo elements —> double colons:
  - ::before
  - ::after

TRANSITIONS, BACKGROUNDS, SVGS

WEEK 4:

TRANSITIONS



TRANSITIONS

- Animation, if done right, catches attention and can even increase user experience
- The easiest way to animate is through CSS transitions:
  CSS property changes from one value to another value over a period of time
- Transitions are applied to pseudo classes like: :hover, :focus, :active
- You can create CSS Transitions with the transition property, a shorthand like margin or padding
- Shorthand of 4 CSS properties:
  - transition-property,
  - transition-duration,
  - transition-timing-function,
  - transition-delay

TRIGGERS

- You can trigger CSS transitions directly with pseudo classes:
  - :hover (activates when mouse goes over an element)
  - :focus (activates when a user tabs onto an element, or when a user clicks into an input element),
  - :active (activates when user clicks on the element)

PROPERTIES

- transition-property,
- transition-duration,
- transition-timing-function,
- transition-delay
- Transition Property:
- These can be used in a comma separated list
- If possible, limit your transitions to transforms and opacity

- List of animatable properties

```
h1 {
    transition: width 1s ease 1s;

    /* is shorthand for */
    transition-property: width;
    transition-duration: 1s;
    transition-timing-function: ease;
    transition-delay: 1s;
}
```

1. TRANSITION-PROPERTY

- Refers to the CSS property you wish to transition
- It is required in the transition shorthand
- You may be tempted to transition every CSS property with all. Don’t ever do this. This is bad for performance. Always specify the property you’re trying to transition.
- If I don’t include it, it will transition all - which as stated above - is bad!

```
#div1 {
        transition-property: width, height, transform;
}
```

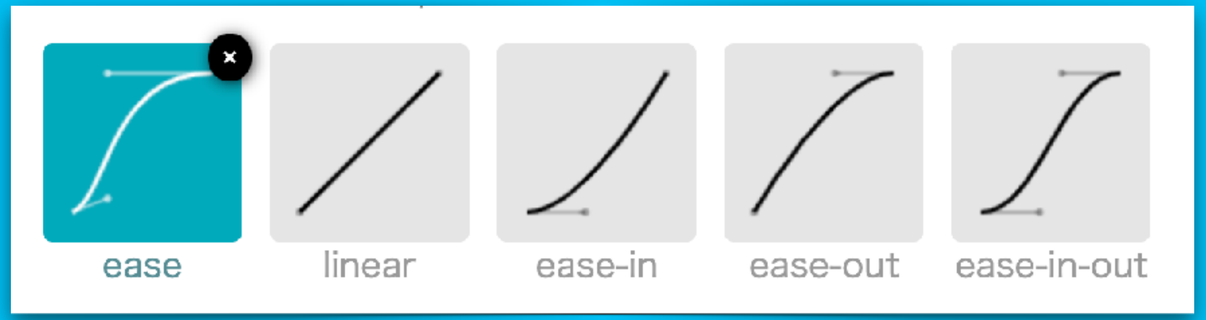
2. TRANSITION-DURATION

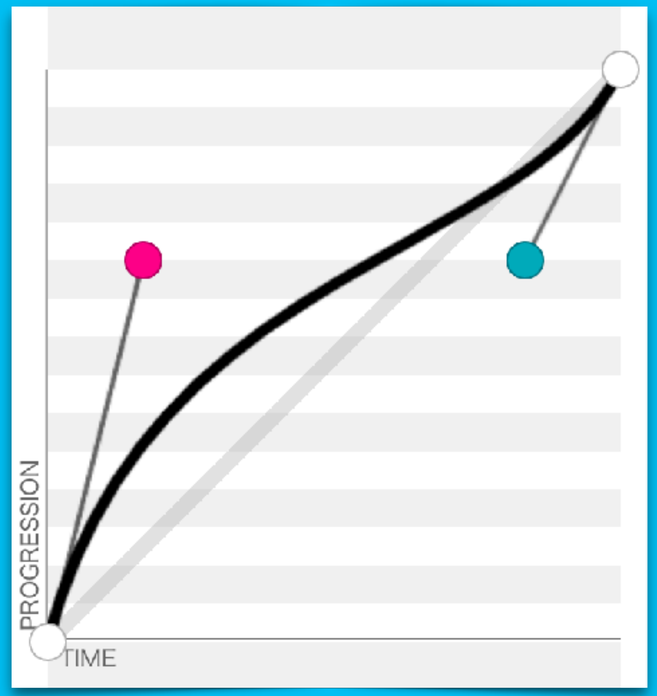
- Refers to the duration of the transition. How long do you want the transition to last? This value is written in seconds with the s or ms suffix: 0.5s aka 500ms
- Better to go with ms because JS only accepts ms
- Default is 0
- It is required in the transition shorthand
- Generally, you want transitions to feel natural. Good measure is between 250ms-400ms

```
#div {
    transition-duration: 350ms;
}
```

3. TRANSITION-TIMING-FUNCTION

- Refers to how the transition occurs and actually gives life to your transitions
- All transitions have a value of linear by default, which means the property changes evenly until the end of the transition
- The thing is, nothing moves linearly in life. That's not how real objects move.
- It is optional in the transition shorthand





3.1. TRANSITION-TIMING-FUNCTION

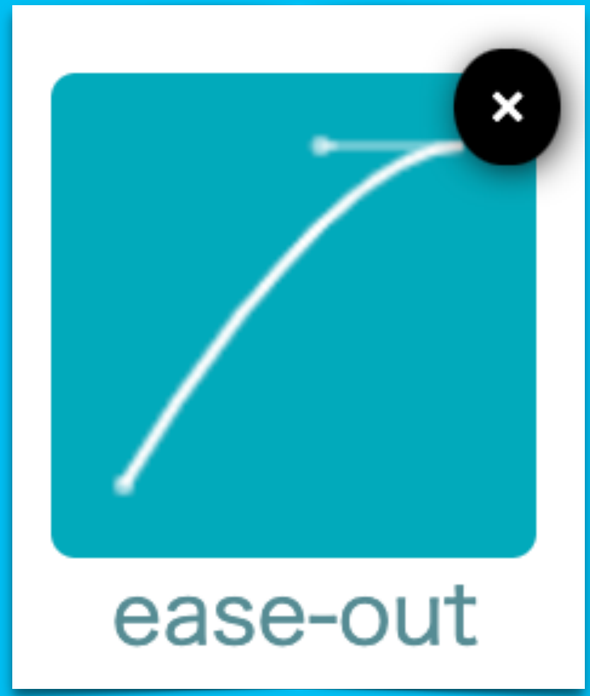
- **Ease-in:**

  Imagine you’re in the car. When you start to move the car, it accelerates slowly and goes toward its top speed.

3.2. TRANSITION-TIMING-FUNCTION

- **Ease-out:**

  Imagine throwing a tennis ball into an open field. The ball leaves your hand with the maximum speed. As it moves, it loses energy, it decelerates and eventually comes to a halt

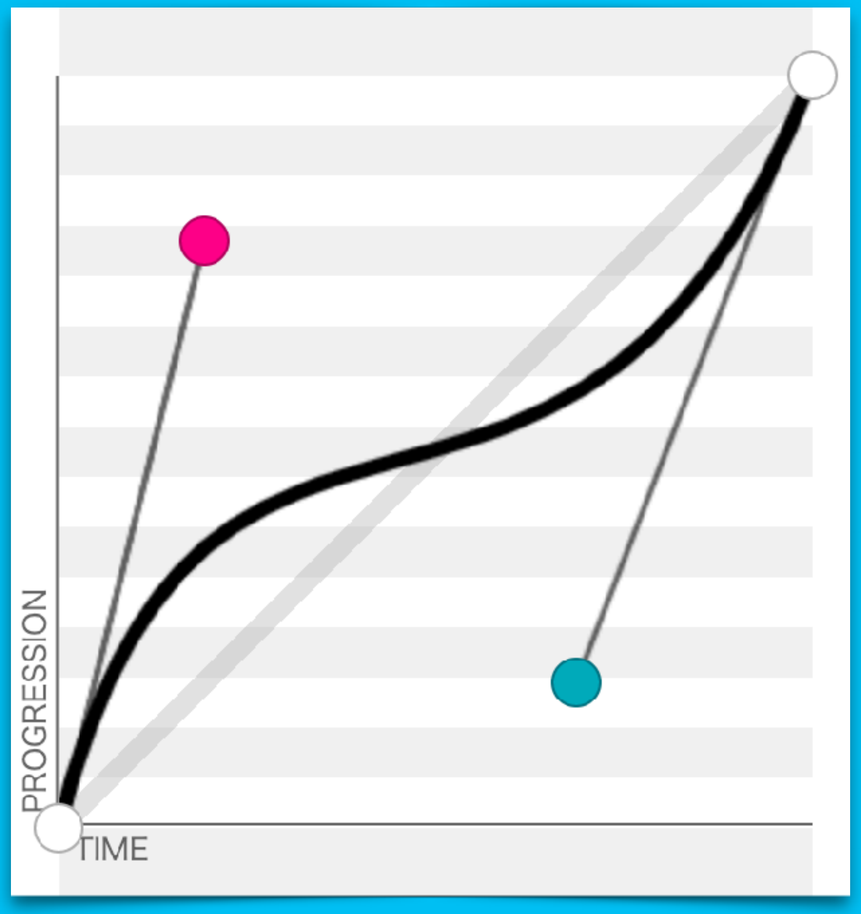


3.3. TRANSITION-TIMING-FUNCTION

- Ease-in-out: There’s also a timing function that combines the two
- I advise against using ease-in-out in your transitions unless your transitions last longer than a second. Nothing eases in and out within a second. It simply looks weird

3.4. TRANSITION-TIMING-FUNCTION

- Cubic-bezier: Lets you create your own timing functions!
- Use your inspector in Chrome/ Firefox to create your own
- Create complex eases with cubic-bezier.com/



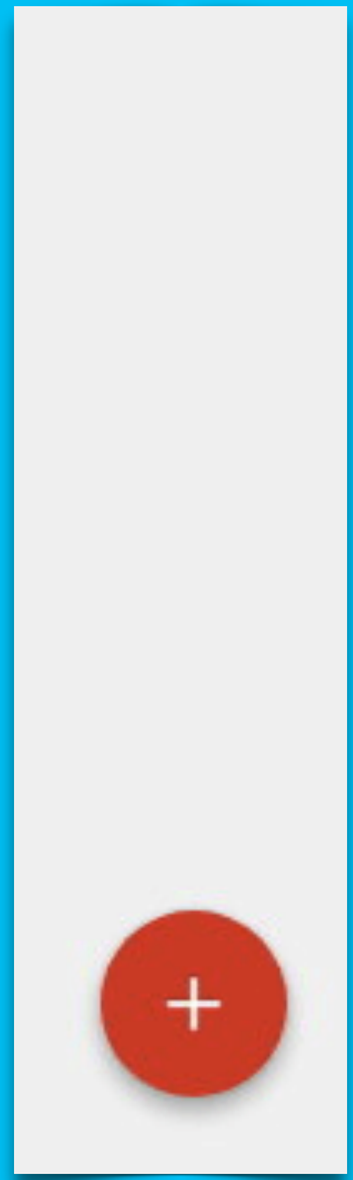
3.5. TRANSITION-TIMING-FUNCTION

- Steps: You decide in how many steps the transition should take place
  - jump-start/start: First jump happens when the transition begins
  - jump-end/end: Last jump happens when the transition ends
  - jump-none: There is no jump on either end
  - jump-both: Includes pauses at both the 0% and 100% marks, effectively adding a step during the transition time.
  - step-start: Equal to steps(1, jump-start)
  - step-end: Equal to steps(1, jump-end)

```
#div1 {
    transition-timing-function: steps(6, end);
}
```

4. TRANSITION-DELAY

- Refers to how long you want to wait before starting the transition
- It is optional in the transition shorthand
- Can be used for simple sequencing
- If you use shorthand, the second number will always refer to the transition delay



5. TRANSITION MORE THAN ONE PROPERTY

- You can transition multiple CSS properties by separating them with a comma in your transition property
- You can do the same with duration, timing-functions and delays as well
- Order is important, as usual

```
#div {
    transition: background-color 500ms ease-out,
                color 350ms ease-in;

    // OR //
    transition-property: background-color, color;
    transition-duration: 500ms, 350ms;
    transition-timing-function: ease-out, ease-in;
}
```

BACKGROUNDS

BACKGROUNDS

- For decor only
- Backgrounds are not accessible. Put images relevant to the content in an img tag!

```
#div1 {
    background-color: transparent;
    background-image: url('path/to/image.png');
    background-repeat: repeat;
    background-position: 0% 0%;
    background-size: auto;
    background-attachment: scroll;
}
```

1. BACKGROUND-COLOR

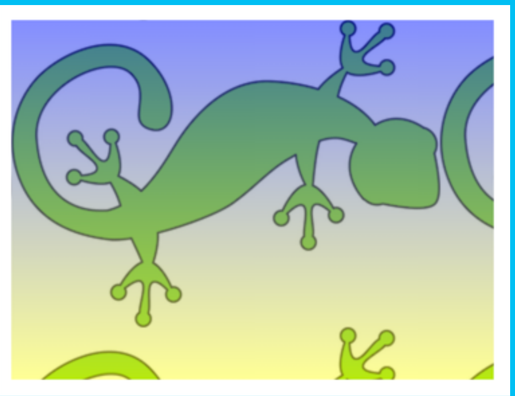
- Background-color property sets the background color of an element
- Accepts the following values:
  - Global (currentcolor)
  - Keyword (red, transparent)
  - Hexadecimal (#11ffeeff)
  - RGBa: rgba(117, 190, 218, 0.5)
  - HSL: hsl(50, 33%, 25%)

```
#div {
    background-color: lightsalmon;
}
```

2. BACKGROUND-IMAGE

- background-image sets one or more background images on an element (aka layering)
- Used for large images, patterns, and gradients
- Use with url(), not with image()
- background-image functions:
  - url()
  - linear-gradient()
  - radial-gradient()
  - repeating-linear-gradient()
  - repeating-radial-gradient()
  - image-set() & more

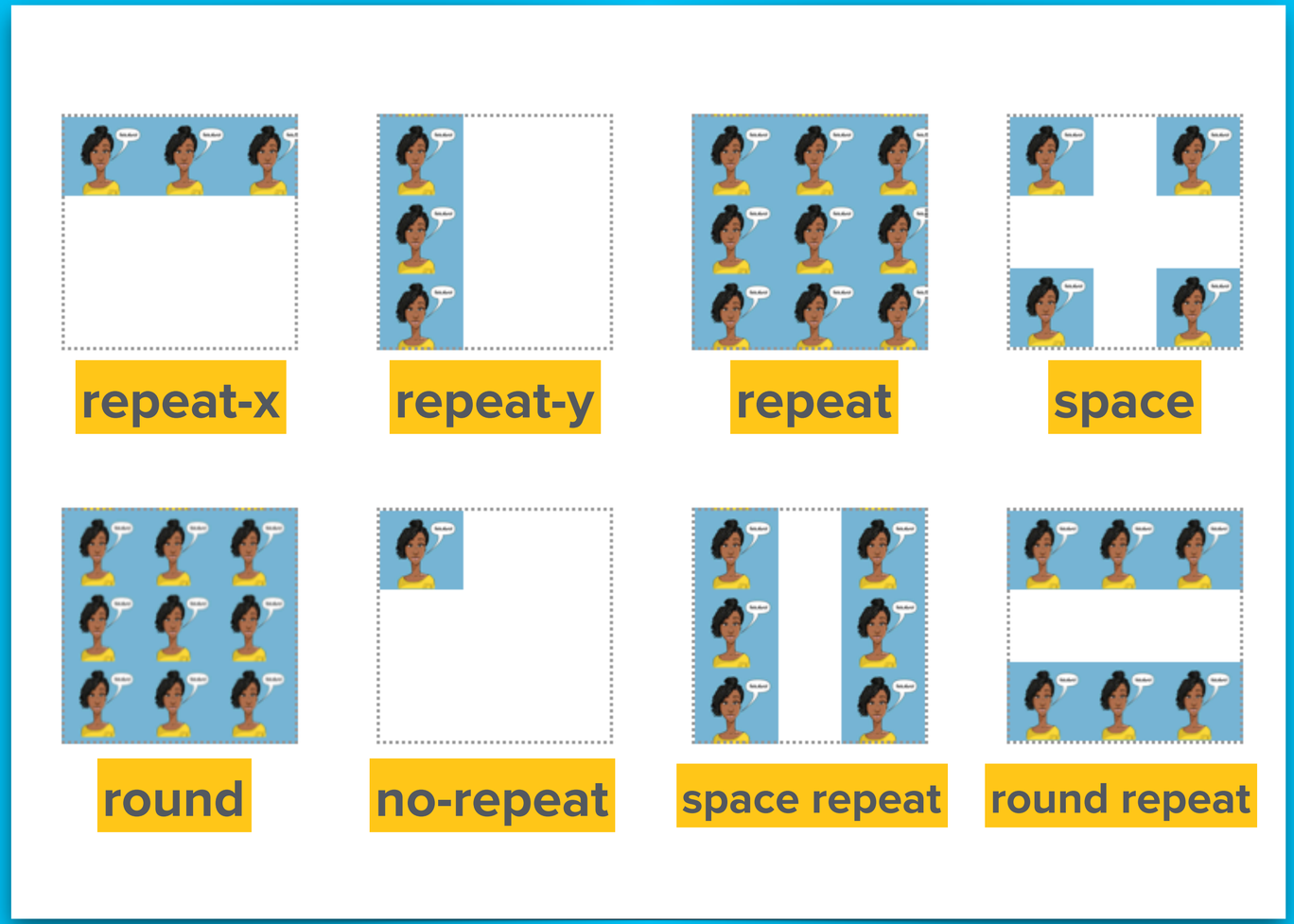
```
#div1 {
    background-image: url("../../media/examples/lizard.png");
}

#div2 {
    background-image: linear-gradient(
            rgba(0, 0, 255, 0.5),
            rgba(255, 255, 0, 0.5)
        ),
        url("../../media/examples/lizard.png");
}
```



3. BACKGROUND-REPEAT

- background-repeat sets how background images are repeated
- A background image can be repeated along the horizontal and vertical axes, or not repeated at all:
  - repeat
  - no-repeat
  - repeat-x
  - repeat-y
  - space
  - round

- Cool effects: https://twitter.com/css/status/839575975478366208



4. BACKGROUND-POSITION

- background-position sets the initial position for each background image.
- The position is relative to the position layer set by background-origin
- Defaults to “top left”
- Lots of position options to choose from!

```
#div1 {
    background-image: url("path/to/image.png");
    background-position: center;
}
#div2 {
    background-position: center right;
}
#div3 {
    background-image: linear-gradient(
        rgba(0, 0, 255, 0.5),
        rgba(255, 255, 0, 0.5)
    );
    background-position: 10% 20%;
}
#div4 {
    background-image: url("path/to/image.png");
    background-position: right 35% bottom 45%;
}
```

5. BACKGROUND-SIZE

- background-size sets the size of the element's background image
- The image can be left to its natural size, stretched, or constrained to fit the available space
- Takes in cover, contain and any length or percentage
- Undeclared values are set to auto
- Be careful to maintain your proportions!

```
#div1 {
    background-size: 200px 400px;
}

#div2 {
    background-size: 100%;
}

#div3 {
    background-size: cover;
}
```

6. BACKGROUND-ATTACHMENT

- background-attachment sets whether a background image's position is fixed within the viewport or scrolls with its containing block
- Values:
  - scroll (default)
  - fixed
  - local

```
#div1 {
    /* Default, scrolls with the page */
    background-attachment: scroll;
}
#div2 {
    /* Relative to viewport, like fixed positioning */
    /* Does not work on mobile!!! */
    background-attachment: fixed;
}
```

ADDITIONAL PROPERTIES

LAYERING

- You can comma separate multiple backgrounds
- Use for textures or effect layering
- http://www.patternify.com/

```
#div1 {
    background-image:
        url('top-image.png'),
        url('next-image.png');
    background-repeat: repeat, no-repeat;
    background-size: auto, cover;
}
```

FAST LOADING

- Load smaller images first and blur them. —> lazy loading!
- filter property applies graphical effects like blur or color shift to an element. Filters are commonly used to adjust the rendering of images, backgrounds, and borders.

```
.big-image {
    background-image: url('smaller-image.png');
    filter: blur(10px);

    &.loaded {
        background-image: url('bigger-image.png');
        filter: blur(0);
    }
}
```

BACKGROUND-BLEND-MODE

- Sets how an element's background images should blend with each other and with the element's background color
- Like Photoshop blend modes

```
#div1 {
    background-color: red;
    background-image: url("/some-img.png");
    background-blend-mode: multiply;
}
#div2 {
    background-blend-mode: darken, luminosity;
}
#div3 {
    background-blend-mode: color-burn;
}
```

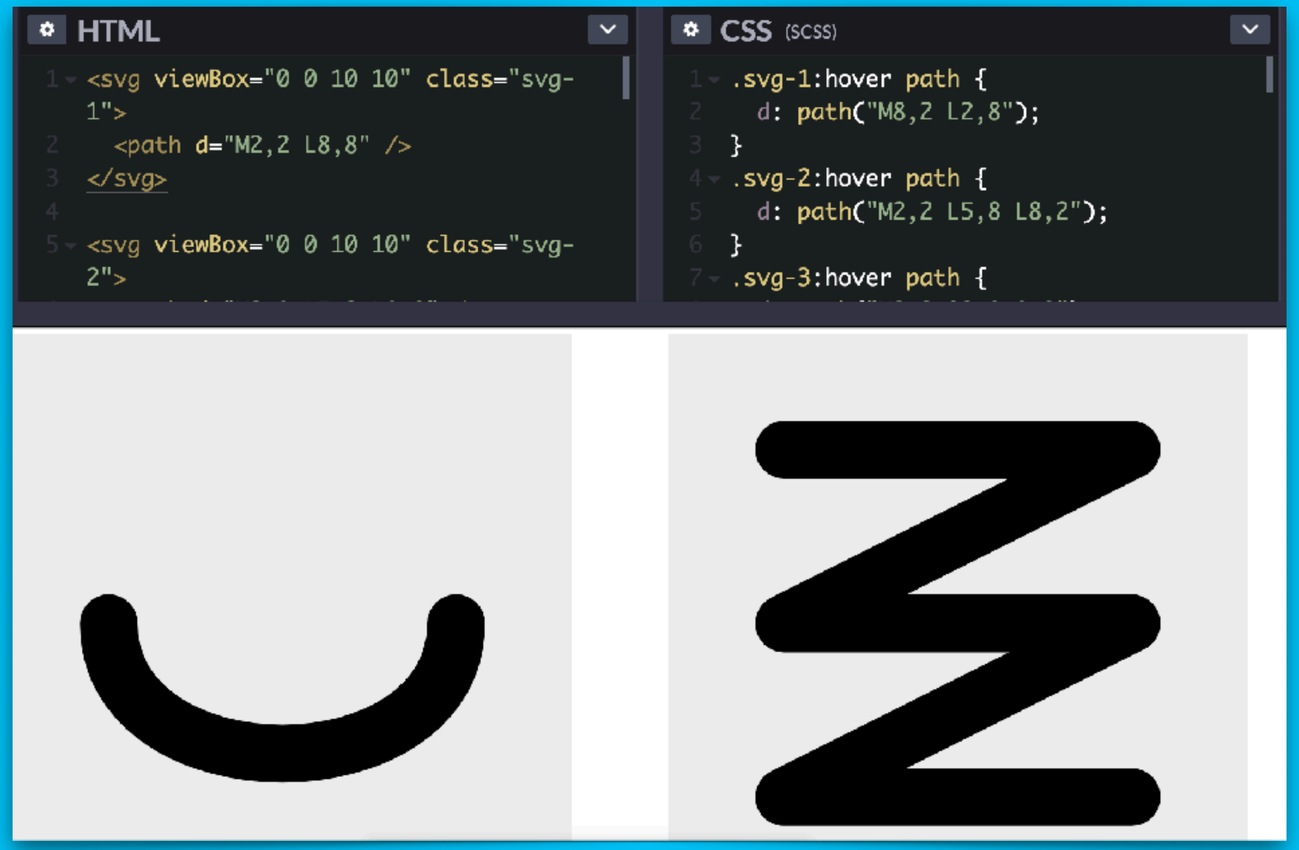
SVGS

SVGS

- Crisp on Retina devices
- Small and compressible file size
- High interactivity potential

- Links:
  - Elastic Menus
  - Dialog Boxes
  - Line Art
  - Practise SVG Manipulation - CodePen

- SVG animations:
  - GSAP



1. IMPLEMENTATION VIA IMAGE

- Simply use in img element like you would with any other image formats
- You will need a width and height attribute for svgs to show
- For non-interactive SVGs, use img with script fallbacks to png version: <img src="your.svg" onerror="this.src='your.png'">
- Pros:
  - The SVG file can be cached by the browser
- Cons:
  - You cannot manipulate the image with JavaScript
  - If you want to control the SVG content with CSS, you must include inline CSS styles in your SVG code
  - You cannot restyle the image with CSS pseudo classes

```
<img
    src="/img/myImage.svg"
    alt="SVG Image"
    width="100"
    height="87"
/>
```

2. IMPLEMENTATION VIA BACKGROUND-IMAGE

- Remember your width and height!

```
.svg-element {
    background-image: url("../images/myImage.svg");
    width: 30vw;
    height: 30vh;
}
```

3. IMPLEMENTATION VIA OBJECT-TAG

- In HTML as well
- Probably the cleanest and most versatile method
- object element represents an external resource, which can be treated as an image, a nested browsing context, or a resource to be handled by a plugin
- You can alter the SVG file itself with CSS & JS
- Use this implementation for your interactive SVGs
- Include a regular img as fallback

```
<object type="image/svg+xml" data="images/myImage.svg">
    Replaceable Internals
    <img src="image.svg" alt="svg image"/>
</object>
```

4. IMPLEMENTATION VIA HTML

- SVG is really just XML
- You can place the code directly inline!
- Again: SVG OMG
- Give it a class or id and style via CSS
- Doesn’t use background-color but fill: <color>
- Makes no image request, easy on CPU
- PITFALLS OF INLINE:
  - Makes a mess of your html file
  - Hard to edit later
  - Harder to cache
  - But better accessibility opportunities

```
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/
svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
viewBox="0 0 200 200" enable-background="new 0 0 200 200"
xml:space="preserve">
    <rect x="5.2" y="5" fill="#ED1F24" width="189.5" height="189.
    5"></rect>
</svg>
```

QUESTIONS?

RESOURCES

- How to use FontAwesome with CSS Pseudo-Elements: https://fontawesome.com/v5.15/how-to-use/on-the-web/advanced/css-pseudo-elements
- Pseudo Element Codepen: https://codepen.io/michellejames/pen/GRNKjGV
- Transition & Background Properties Codepen: https://codepen.io/michellejames/pen/XWNrNQw
- Cubic Bezier: https://cubic-bezier.com/#.17,.67,.83,.67
- SVG Paths Codepen: https://codepen.io/michellejames/pen/bGBbrZp

HOMEWORK - PROJECT 1

- Expand upon the product page you made in Week 2.
- Make sure to create a new branch for your changes here, something like “Added xy styling component”.
- You must add or alter content to demonstrate what we learned about transitions and backgrounds. At a minimum:
- Use a background image in a creative way. The simplest route to go would be layering multiple background images or using blend modes to tint one. Take a shot at using a property you haven't used before.
- Use CSS transitions to create an animation sequence that reveals/hides some of your page's content. You should use a hover state or class name change to trigger transitions on multiple elements. This can be anything from a nav bar expanding to complex decorative effects or revealing more text when hovering over each price tier. Also note that this means no CSS keyframe animations, and no javascript animation. This is intended to demonstrate that you have solid control over the transition properties.
- Add some SVG icons, whether they are social media icons or others used to enhance your price tier designs. Look to FontAwesome or similar for sourcing some generic and well-optimized icons.
- As usual, the page must be fully responsive. I'd also like to see usage of Sass and BEM.
- Submit a link to your GitHub repo.

FIN