A. Images for web
    a. Vector images
        i. use lines, points, and polygons to represent an image
        ii. essentially giant math equations, and each dot, line and shape is represented by its own equation
        iii. ideal for images that consist of geometric shapes
        iv. zoom and resolution-independent
            1. Scaling up a vector = smooth, crisp graphics
            2. ideal format for high-resolution screens and assets that need to be displayed at varying sizes
        v. What do I mean by high-resolution screens?
            1. High resolution screens have multiple device pixels per CSS pixel
                a. CSS pixel does not always equal device pixel
                b. Single CSS pixel may contain multiple device pixels
                c. The more device pixels there are, the finer the detail of the displayed content on the screen
            2. High resolution images require significantly higher number of pixels and bytes
                a. High DPI screens (dots per inch = pixels fit into an inch)
                b. image assets need more detail in order to use higher device pixel counts
                c. vector images = rendered at any resolution with sharp results
                d. raster images = data on a per-pixel basis
                    i. larger the number of pixels = the larger the filesize of a raster image
                    ii. photo asset displayed at 100x100 (CSS) pixels: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization
                    iii. double the resolution of the physical screen = quadruples the number of required pixels
                e. high resolution screens also require high-resolution images
                    i. prefer vector images whenever possible
                    ii. deliver and optimize multiple variants of each image (will talk about responsive images with responsive layouts)
        vi. SVG
            1. Stands for Scalable Vector Graphics
            2. XML-based markup language for describing two dimensional based vector graphics
                a. essentially to graphics what HTML is to text
                b. defined in XML text files which can be searched, indexed, scripted and compressed
                c. can be created and edited with any text editor and with drawing software
                d. Designers will generally create/edit SVGs for you, but it helps to know what an SVG file looks like
            3. Example: SVG in HTML file
            4. SVG editor: https://svg-edit.github.io/svgedit/releases/latest/editor/svg-editor.html
            5. Include as img or background-image: if you don't need to modify with CSS
            6. Include as inline: if you want to access inner paths/shapes with CSS/scripting/filters

7. Include as data uri: if you don't need to modify and you want to save additional data file download

b. Raster images
    i. represent an image by encoding the individual values of each pixel within a 2-dimensional grid of individual "pixels"
        1. 100x100 pixel image is a sequence of 10,000 pixels
        2. each pixel stores the "RGBA" values: (R) red channel, (G) green channel, (B) blue channel, and (A) alpha (transparency) channel
    ii. complex scenes with lots of irregular shapes and details
    iii. Scaling up a raster image = jagged and blurry graphics
    iv. Lossless vs lossy image compression
        1. For certain data like source code or executable
            a. critical that a compressor does not alter or lose any of the original information
            b. single missing or wrong bit of data could completely break it entirely
            c. "Lossless" compression = capture all of the data of your original file
                i. Nothing from the original file is lost
                ii. file may still be compressed
                iii. all lossless formats will be able to reconstruct file to its original state
        2. For some other types of data, such as images, audio, and video
            a. may be perfectly acceptable to deliver an "approximate" representation of the original data.
            b. can often get away with discarding some information about each pixel in order to reduce the filesize of an image
            c. "Lossy" compression = approximate what your original image looks like
                i. might reduce the amount of colors in your image or analyze the image for any unnecessary data
                ii. typically reduce the file size, though they may reduce the quality of your image
        3. Lossy files typically much smaller than lossless files
        4. Each raster image file is either lossless or lossy, depending on how the format handles your image data
            a. any image can undergo a lossy compression step to reduce its size
            b. difference between various image formats GIF, PNG, JPEG, and others = combination of the specific algorithms they use or not when applying the lossy and lossless steps
            c. No optimal configuration/universal setting
                i. All depends on the image and what is acceptable to the display
    v. GIF
        1. lossless raster format
        2. stands for Graphics Interchange Format
        3. Limited to 256 colors
        4. Supports alpha channel transparency
            a. Aliased transparency
            b. "Jaggy edge", transparency does not blend nicely
    vi. PNG
        1. lossless raster format

2. stands for Portable Network Graphics
3. “next-generation GIF”
4. Supports anti-aliased transparency
    a. “Smooth edge”, blends nicely
5. can display higher color depths --> translates into millions of colors
6. web standard, quickly becoming one of the most common image formats used online

vii. JPG

1. lossy raster format
2. stands for Joint Photographic Experts Group (technical team that developed it)
3. have a sliding scale of compression that decreases file size tremendously, but increases artifacts or pixelation the more the image is compressed

B. Background images
    a. applies an image or gradient to the background of an element
    b. Images
        i. `Background-image: url(foo.jpg);`
        ii. url() -- file path to any image to use for background
            1. Relative: relative to the folder from which css is loaded
                a. if images are in another folder, use absolute path or relative to the root path (starting with /)
            2. Absolute
        iii. Multiple background images: `background-image: url(image-front.jpg), url(image-back.jpg)`
            1. `Comma separated values`
        iv. Image will tile horizontally and vertically by default
            1. `No-repeat`
            2. `Repeat-x`
            3. `Repeat-y`
            4. `Comma separated values for each background image`
    c. Gradients
        i. Linear -- straight line
            1. `background-image: linear-gradient(red, green);`
            2. Top to bottom by default
                a. “To” keyword to change direction
                    i. `background-image: linear-gradient(to bottom right, red, green);`
                b. Degrees
                    i. `background-image: linear-gradient(135deg, red, green);`
                    ii. 0deg = vertical gradient running bottom to top
                    iii. 90deg = horizontal gradient running left to right
                    iv. 180deg = vertical gradient running top to bottom
                    v. 270deg = horizontal gradient running right to left
                    vi. clockwise direction
                    vii. negative angles run in the counterclockwise direction
            3. Color stop: change from one color to another at specific spot
                a. `background-image: linear-gradient(red, green 25%);`

        b. two color stops that are the same = color instantly changes to another solid color
            i. `background-image: linear-gradient(red, red 25%, green 25%);`
    ii. Radial -- circle out from point
        1. `background-image: radial-gradient(red, green);`
        2. Default shape is ellipse if box is not square
            a. Force to circle: `background-image: radial-gradient(circle, red, green);`
        3. Location
            a. "At" keyword to change location of center
                i. `background-image: radial-gradient(at top right,` red, green);
        4. Supports linear concepts like color stops, multiple colors
    iii. Conical
        1. similar to a radial gradient
            a. both are circular
            b. Both use the center of the element as the source point
            c. Radial = colors emerge from the center of the circle
            d. conic = colors move around the circle
            e. Look like the shape of a cone from above
            f. Starts at top (0), moves clockwise around
        2. Supports linear concepts like color stops, multiple colors
        3. Not supported in anything but Chrome basically

C. Effects
    a. Shadows
        i. Box-shadow: Used in casting shadows (drop shadows)
            1. Eg: `box-shadow: 5px 5px 10px 10px red;`
            2. 5 parameters
            3. The horizontal offset (required)
                a. positive = the shadow will be on the right of the box
                b. negative = the shadow on the left of the box
            4. The vertical offset (required)
                a. negative = the shadow will be above the box
                b. positive = the shadow will be below the box
            5. The blur radius (required),
                a. 0 the shadow will be sharp
                b. the higher the number, the more blurred it will be, and the further out the shadow will extend
            6. The spread radius (optional)
                a. positive = increase the size of the shadow
                b. negative = decrease the size of the shadow
                c. Default is 0 (the shadow is same size as its blur)
            7. Color (required)
                a. If color is omitted, shadow will use are drawn in the foreground text color
                b. Can take any named value, hex, rgb, rgba
            8. Shadow can be inset into the box as well
                a. "Inset" keyword

        b. `Box-shadow: inset 5px 5px 10px 10px red;`
    9. Can comma separate box shadows to apply multiple shadows around box
b. Animations
    i. Ability to animate transitions from one CSS style configuration to another
    ii. consist of two components
        1. style describing the CSS animation
        2. set of keyframes that indicate the start and end states of the animation's style + possible intermediate points
    iii. Advantages to CSS animations:
        1. easy to use for simple animations (does not require knowing JS)
        2. the animations run well
            a. browser can use frame-skipping and other techniques to keep the performance as smooth as possible
        3. the browser optimizes performance and efficiency
    iv. Animation sub properties:
        1. Animation-name: the name of the @keyframes describing the animation's keyframes
        2. Animation-duration: length of time that an animation should take to complete one cycle
        3. Animation-timing-function: how the animation transitions through keyframes, by establishing acceleration curves (linear, ease-in, ease out, etc.)
            a. https://developer.mozilla.org/en-US/docs/Web/CSS/animation-timing-function
        4. Animation-delay: delay between the time the element is loaded and the beginning of the animation sequence
        5. Animation-iteration-count: number of times the animation should repeat
            a. Use "infinite" to repeat forever
        6. Animation-direction: should the animation alternate direction on each run through the sequence or reset to the start point and repeat itself
        7. Animation-fill-mode: what values are applied by the animation before and after it is executing
            a. Does it retain the last css styles to persist the change? Does it reset itself?
            b. https://developer.mozilla.org/en-US/docs/Web/CSS/animation-fill-mode
        8. Animation-play-state: pause and resume the animation sequence.
    v. Example
D. Optimizing images
    a. Do you need an image at all?
        i. Eliminate unnecessary image resources
            1. Every image downloaded = extra data load on your user
        ii. Select the right image format
            1. three universally supported image formats: GIF, PNG, and JPEG
            2. Need small file sizes for simple graphics?
                a. Use GIF
                b. compression techniques in the GIF format allow image files to shrink tremendously
            3. need animation?
                a. GIF is the only universal choice

    b. GIF limits the color palette to at most 256 colors
4. Need transparency?
    a. PNG = alpha transparency
    b. GIF = aliased transparency
5. need to preserve fine detail with highest resolution?
    a. Use PNG
        i. does not apply any lossy compression algorithms beyond size of the color palette
        ii. highest quality image, but at a cost of significantly higher filesize than other formats -- use wisely!
6. Image is a photo, screenshot, or similar?
    a. Use JPEG
    b. JPEG uses a combination of lossy and lossless optimization to reduce file size of the image asset
    c. Try different JPEG quality levels to find the best quality vs. filesize tradeoff for your asset
7. image contains imagery composed of geometric shapes?
    a. consider converting it to SVG format
8. Image contains text?
    a. Use web fonts instead of encoding text in images
    b. Still selectable, searchable, resizeable, easily translatable -- usability!

iii. SVG files should be minified/optimized to reduce their size
1. https://jakearchibald.github.io/svgomg/
2. Illustrator SVG → optimized

iv. Compress raster images
1. Bit depth
    a. number of memory bits used to store color data for each pixel in a raster image
        i. 1 bit = 2 colors (black and white) "bit map"
        ii. 2 bits = 4 colors
        iii. 4 bits = 16 colors
        iv. 8 bits = 256 colors
        v. 16 bits = "thousands"
        vi. 32 bits = "millions"
    b. reduce the "bit-depth" of the image from 8 bits per channel to a smaller color palette
        i. 8 bits per channel = 256 values per channel = 16,777,216 (256 ^ 3) colors in total
        ii. Reducing to 256 colors = 8 bits in total for the RGB channels and immediately save two bytes per pixel
    c. Complex scenes with gradual color transitions (gradients, sky, etc.) require larger color palettes to avoid visual artifacts
    d. if the image only uses a few colors, then a large palette is simply inflating bits for no reason
2. Leverage CSS3 effects (shadows, gradients, etc.) where possible
3. Already being loaded along with your css
4. CSS effects/animations
    a. produce resolution-independent assets

b. always look sharp at every resolution and zoom level
c. fraction of the bytes required by an image file