A. CSS media queries
    a. simple filters that can be applied to CSS styles
    b. Eg `@media (min-width: 800px) { /* styles go here */ }`
    c. Media queries let us present the same HTML content as distinct CSS layouts
    d. make it easy to change styles based on the characteristics of the device like display type, width, height, orientation, resolution
    e. A media query computes to true when the media type (if specified) matches the device on which a document is being displayed and all media feature expressions compute as true
        i. If true, then all the styles nested inside the query are applied (specificity rules apply!)
        ii. Queries involving unknown media types are always false
        iii. Queries where at least one of the expressions compute as false are false
    f. Media types
        i. general category of a device
        ii. If using not or only logical operators, the media type is required
        iii. Otherwise, media type is implied to be all
        iv. All = Suitable for all devices
        v. Print = Intended for paged material and documents viewed on a screen in print preview mode
        vi. Screen = Intended primarily for screens
        vii. Speech = Intended for speech synthesizers
    g. Media queries to apply styles based on device characteristics
        i. min-width: browser width greater than the value defined in the query
        ii. max-width: browser width less than the value defined in the query
        iii. min-height: browser height greater than the value defined in the query
        iv. max-height: browser height less than the value defined in the query
        v. orientation=portrait: browser where the height is greater than or equal to the width
        vi. orientation=landscape: browser where the width is greater than the height
    h. Examples
        i. Background colors example
        ii. Grid content example
B. Feature queries
    a. @supports lets you specify declarations that depend on a browser's support for one or more specific CSS features
    b. Not supported in IE 11, is supported in Edge
C. Breakpoints
    a. a point that allows for the provision of the best possible layout that enables users to consume or understand your site's content for their current viewport size
    b. Create breakpoints based on content, not devices
        i. Devices change, responsive/adaptive development should ideally work across all
        ii. Breakpoints are easier to work around content than layout/design
        iii. Keep lines of text to max 70-80 chars
        iv. Optimize for reading
            1. As columns become too wide for comfortable reading/display, add breakpoints
            2. Use adaptive font sizing (ems) at different viewport widths
    c. If you're following mobile first, you start with the smallest screen width and then work your way up
        i. As you expand that view, there will be a point at which the layout looks terrible
        ii. that's where you add a breakpoint!

d. Don't hide content!
    i. If content doesn't fit at certain breakpoints, is it necessary?
e. Don't target devices!
    i. Set breakpoints wherever the presentation of the content degrades instead of specific device widths
    ii. This covers any device that comes along later
    iii. 2 methods for breakpoint development
        1. "Progressive enhancement"
            a. Build from bottom up
            b. Mobile first = content first
                i. Developing within mobile parameters forces you to prioritize content
                ii. Content focus = user focus
        2. "graceful degradation"
            a. build from top down
            b. Problem w/graceful degradation is that it treams mobile design as an afterthought and makes you think about the mobile experience as "cutting down" the "real" experience rather than an integral experience itself
f. Example: Edgar Allen Poe article w/breakpoints

D. Viewport
a. Using meta tag controls the width and scaling of the browser's viewport
b. Eg `<meta name="viewport" content="width=device-width, initial-scale=1">`
    i. The width property controls the size of the viewport
        1. It can be set to a specific number of pixels like width=600 or to the special value device-width, which is the width of the screen in CSS pixels at a scale of 100%
        2. instructs the page to match the screen's width in device-independent pixels
        3. allows the page to reflow content to match different screen sizes
        4. "Device-independent": units that provide a flexible way to accommodate a design across platforms
    ii. Initial-scale controls the zoom level when the page is first loaded
        1. It establishes a 1:1 relationship between CSS pixels and device-independent pixels
c. Relative sizes for elements to avoid breaking layout
    i. Key concept of responsive design = fluidity and proportionality as opposed to fixed width layouts
    ii. Your layout isn't really responsive/adaptive if you're specifying all your dimensions in absolute pixels
    iii. Relative units for measurements
        1. simplify layouts
        2. prevent accidental elements that are too big for the viewport
        3. allows browsers to render the content based on the user's zoom level
        4. Eg width
            a. 100% on div = spans the width of the viewport
            b. never too big or too small for the viewport
            c. fits, no matter the device
    iv. Flexbox and grid are responsive by default
        1. Use these to simplify responsive layouts!

v. Using rems/ems/px for media queries
    1. Media queries use default font size 16px for rem/em, not whatever you have declared in your css!

E. Responsive images
    a. Desktop size/shape images sometimes don't work for mobile size & vice versa so there are a few ways to make images responsive to viewport widths
    b. CSS solutions
        i. Resizing
            1. Don't need to worry about swapping out images
            2. `Width: 100%; height: auto;`
                a. image can be scaled up to be larger than its original size
                b. Ok for vector images which can zoom/resize without becoming pixelated, not so good for raster images
            3. `max-width: 100%, height: auto;`
                a. the image will scale down if it has to, but never scale up to be larger than its original size
            4. `Background: url(foo.jpg); background-size: 100% 100%;`
                a. the background image will stretch to cover the entire content area
            5. `Background: url(foo.jpg); background-size: cover;`
                a. the background image will scale to cover the entire content area. Notice that the "cover" value keeps the aspect ratio, and some part of the background image may be clipped
            6. `Background: url(foo.jpg); background-size: contain;`
                a. The background image will scale to one dimension covering the content area, but will not scale in a way that distorts the image
        ii. Resolution switching
            1. wanting to show the identical image displayed to suit different resolutions in order to take advantage of different user bandwidth levels
                a. Eg. You don't want to serve full size high resolution images to mobile
            2. The ideal situation is to have the same image available at multiple resolutions and serve the appropriate size depending upon the device
            3. We use an HTML solution to swap out higher/lower resolution images rather than resizing with CSS
                a. Raster images look grainy when the viewport is so big that the image is displayed at sizes larger than original
                b. Vector images don't have this problem, but remember that they are not able to support photorealistic images, so they are not ideal for every image situation
            4. HTML to resolution switch images for the same image, different dimensions, same screen resolution
                a. Srcset = Set of images the browser can choose from to display
                    i. Comma separated list of image filename + image's actual/inherent width
                    ii. Eg `srcset="foo.jpg 400w, bar.jpg 600w, baz.jpg 800w"`
                b. Sizes
                    i. Set of media conditions (eg, screen widths) + width of the slot that the image will fill when the condition is true

ii. For the slot width, absolute length (px, em) or a length relative to the viewport (vw), but not percentages
iii. Last slot width has no media condition — this is the default chosen if no other conditions match
iv. Browser ignores everything after the first matching condition

c. Src = older browsers that don't support these features will ignore them and load src image
d. Browser process
i. Look at its device width
ii. Work out which media condition in the sizes list is the first one to be true
iii. Look at the slot size given to that media query
iv. Load the image referenced in the srcset list that most closely matches the chosen slot size

5. HTML to resolution switch images for the same image, same dimensions, different screen resolutions (eg, retina vs. standard)
a. Eg `<img srcset="foo.jpg 1x, bar.jpg 1.5x, baz.jpg 2x" src="baz.jpg" alt="This is an image">`
b. Srcset = Set of images the browser can choose from to display
i. Comma separated list of image filename + x descriptor of image resolution as multiplier of standard (1x, 2x, etc.)
ii. Eg `srcset="foo.jpg 1x, bar.jpg 1.5x, baz.jpg 2x"`
c. Sizes: none, browser just chooses appropriate resolution
d. Src = older browsers that don't support these features will ignore them and load src image
e. Browser process
i. Look at its device resolution
ii. Load the image referenced in the srcset list that most closely matches the chosen resolution

iii. Image switching
1. wanting to change the image displayed to suit different image display sizes
2. Images at different scale/size might look weird/less appropriate
3. HTML solutions
4. Picture & source elements
a. Media: media conditions (eg, screen widths)
i. Use media attribute only when swapping completely different images
ii. If using media, don’t also include a sizes attribute
b. Srcset: same as above (could have comma separated list)
i. Eg <source media="(max-width: 799px)" srcset="foo-cropped-one-way.jpg">
c. Img
i. you must provide an img element, with src and alt
ii. Must be right before </picture>
iii. otherwise no images will appear

iv. Why not CSS/JS?
1. Browser preloads images before the main parser loads CSS and JavaScript

2. If you load the <img> element, then detect the viewport w/JS and dynamically change the source image to a smaller
3. the original image would already have been loaded
4. would load the small image as well
5. Multiple image downloads