A. Viewport
    a. Mobile browsers try to optimize experience for user
        i. Tries to render the page at a desktop screen width
            1. usually about 980px, varies by device
        ii. Tries to make the content look better by increasing font sizes and scaling the content to fit the screen
        iii. For sites not responsive, this is good
            1. Site will be scaled down so that it's viewable
        iv. For sites not responsive, this is not so good
            1. If your site is limited to eg 1024px on an ipad and you have breakpoints set for other widths, they will not be activated
    b. Using meta tag controls the width and scaling of the browser's viewport.
        i. Eg `<meta name="viewport" content="width=device-width, initial-scale=1">`
            1. The width property controls the size of the viewport
                a. 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%
                b. instructs the page to match the screen's width in device-independent pixels
                c. allows the page to reflow content to match different screen sizes
                d. "Reset.css" for screen size content
                e. "Device-independent": units that provide a flexible way to accommodate a design across platforms
                    i. screen pixel density and resolution vary depending on the platform
                    ii. Eg Mac retina display=220 ppi
                    iii. "Normal" display usually between 100-150 ppi
            2. Initial-scale controls the zoom level when the page is first loaded
                a. establishes a 1:1 relationship between CSS pixels and device-independent pixels
                    i. some browsers keep the page's width constant when rotating to landscape mode
                        1. zoom rather than reflow to fill the screen
                    ii. allows the page to take advantage of the full landscape width when switching orientation
        ii. Other attributes
            1. minimum-scale
            2. maximum-scale
            3. user-scalable
            4. when set, can disable the user's ability to zoom the viewport
                a. potentially causing accessibility issues
                b. Best practice: don't set these
B. CSS media queries
    a. Definition
        i. simple filters that can be applied to CSS styles
        ii. make it easy to change styles based on the characteristics of the device
            1. display type
            2. width

3. height
4. orientation
5. resolution

b. What does a media query look like?
    i. Eg `@media (min-width: 800px) { /* styles go here */ }`
    ii. 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
        1. If true, then all the styles nested inside the query are applied
        2. Queries involving unknown media types are always false
        3. Queries where at least one of the expressions compute as false are false
    iii. Media types
        1. general category of a device
        2. If using not or only logical operators, the media type is required
        3. Otherwise, media type is implied to be all
            a. All = Suitable for all devices
            b. Print = Intended for paged material and documents viewed on a screen in print preview mode
            c. Screen = Intended primarily for screens
            d. Speech = Intended for speech synthesizers
c. Media queries to apply styles based on device characteristics
    i. Most used
        1. min-width: browser width greater than the value defined in the query.
        2. max-width: browser width less than the value defined in the query.
        3. min-height: browser height greater than the value defined in the query.
        4. max-height: browser height less than the value defined in the query.
        5. orientation=portrait: browser where the height is greater than or equal to the width
        6. orientation=landscape: browser where the width is greater than the height.
        7. Any filter meeting that criteria = resulting CSS block is applied (precedence rules apply)
    ii. Examples
        1. Background colors example
        2. Grid content example
d. Relative sizes for elements to avoid breaking layout
    i. Key concept of responsive design = fluidity and proportionality as opposed to fixed width layouts
    ii. 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
    iii. Eg width
        1. 100% on div = spans the width of the viewport
        2. never too big or too small for the viewport
        3. fits, no matter the device
    iv. Flexbox and grid are responsive by default
        1. Use these to simplify responsive layouts!
e. Using rems/ems/px for media queries

        i. Media queries use default font size 16px for rem/em, not whatever you have declared in your css
        ii. Remember when calculating media query widths
C. Feature queries
    a. @supports
        i. lets you specify declarations that depend on a browser's support for one or more specific CSS features
        ii. Not supported in IE 11, is supported in Edge
D. Responsive developer tools
    a. Chrome developer tools
    b. Device toggle
    c. Image widths
E. Responsive images
    a. Desktop size/shape images sometimes don't work for mobile size & vice versa
    b. Simple resizing: css solutions
        i. Don't need to worry about swapping out images
        ii. Width: 100%; height: auto;
            1. image can be scaled up to be larger than its original size
            2. Ok for vector images which can zoom/resize without becoming pixelated, not so good for raster images
        iii. max-width: 100%, height: auto;
            1. the image will scale down if it has to, but never scale up to be larger than its original size
        iv. Background: url(foo.jpg); background-size: 100% 100%;
            1. the background image will stretch to cover the entire content area
        v. Background: url(foo.jpg); background-size: cover;
            1. 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
        vi. Background: url(foo.jpg); background-size: contain;
    c. Resolution switching
        i. wanting to show the identical image displayed to suit different resolutions
        ii. Raster images look grainy when displayed at sizes larger than original
        iii. Vector images are not able to support photorealistic images
        iv. Bandwidth restrictions
            1. You don't want to serve full size high resolution images to mobile
        v. Ideal situation
            1. multiple resolutions available
            2. serve the appropriate size depending upon the device
        vi. HTML solutions -- same resolution
            1. Eg `<img srcset="foo.jpg 400w, bar.jpg 600w, baz.jpg 800w" sizes="(max-width: 320px) 280px, (max-width: 600px) 440px, 800px" src="baz.jpg" alt="This is an image">`
                a. Srcset
                    i. Set of images the browser can choose from to display
                    ii. Comma separated list of image filename + image's actual/inherent width

iii. 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

2. Browser process
   a. Look at its device width.
   b. Work out which media condition in the sizes list is the first one to be true.
   c. Look at the slot size given to that media query.
   d. Load the image referenced in the srcset list that most closely matches the chosen slot size.

vii. HTML solutions -- different resolutions (eg, retina vs. standard)

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

d. Image switching
   i. wanting to change the image displayed to suit different image display sizes
   ii. Images at different scale/size might look weird/less appropriate
   iii. HTML solutions
      1. Eg `<picture><source media="(max-width: 799px)" srcset="foo-cropped-one-way.jpg"><source media="(max-width: 1200px)" srcset="foo-cropped-another-way.jpg"><img src="foo.jpg" alt="This is a picture"></picture>`
         a. Source
            i. Media: media conditions (eg, screen widths)
               1. Use media attribute only when swapping completely different images
               2. If using media, don’t also include a sizes attribute
            ii. Srcset: same as above (could have comma separated list)

iii. Eg `<source media="(max-width: 799px)" srcset="foo-cropped-one-way.jpg">`

b. Img
   i. you must provide an img element, with src and alt
   ii. Must be right before </picture>
   iii. otherwise no images will appear

e. Why not CSS/JS?
   i. Browser preloads images before the main parser loads CSS and JavaScript
   ii. If you load the <img> element, then detect the viewport w/JS and dynamically change the source image to a smaller
      1. the original image would already have been loaded
      2. would load the small image as well
      3. Multiple image downloads