Basic principles of responsive web design

- Mobile-first
- Progressive enhancement
- Fluid layouts
- Media Queries
- Images
- Typography

Mobile First

- Different user **goals** - more focused
- Different user **emotions** - more personal
- Different user **cognitive approach** - less attention, if possible. While watching TV, "second screen”. Hopefully not while driving.
- Different user **environment** - more intimate locations, more variety (shopping, transit, work, home)
- Different user **time** - lots of use at commute time and at night for mobile but not desktop
- Different user **interface**
    - Fat finger, gestural, accelerometer, camera, location-based
    - Wide variations in device resolution, viewport size

How to mobilize

- From http://www.nngroup.com/articles/mobile-site-vs-full-site/

- **cut features**, to eliminate things that are not core to the mobile use case;

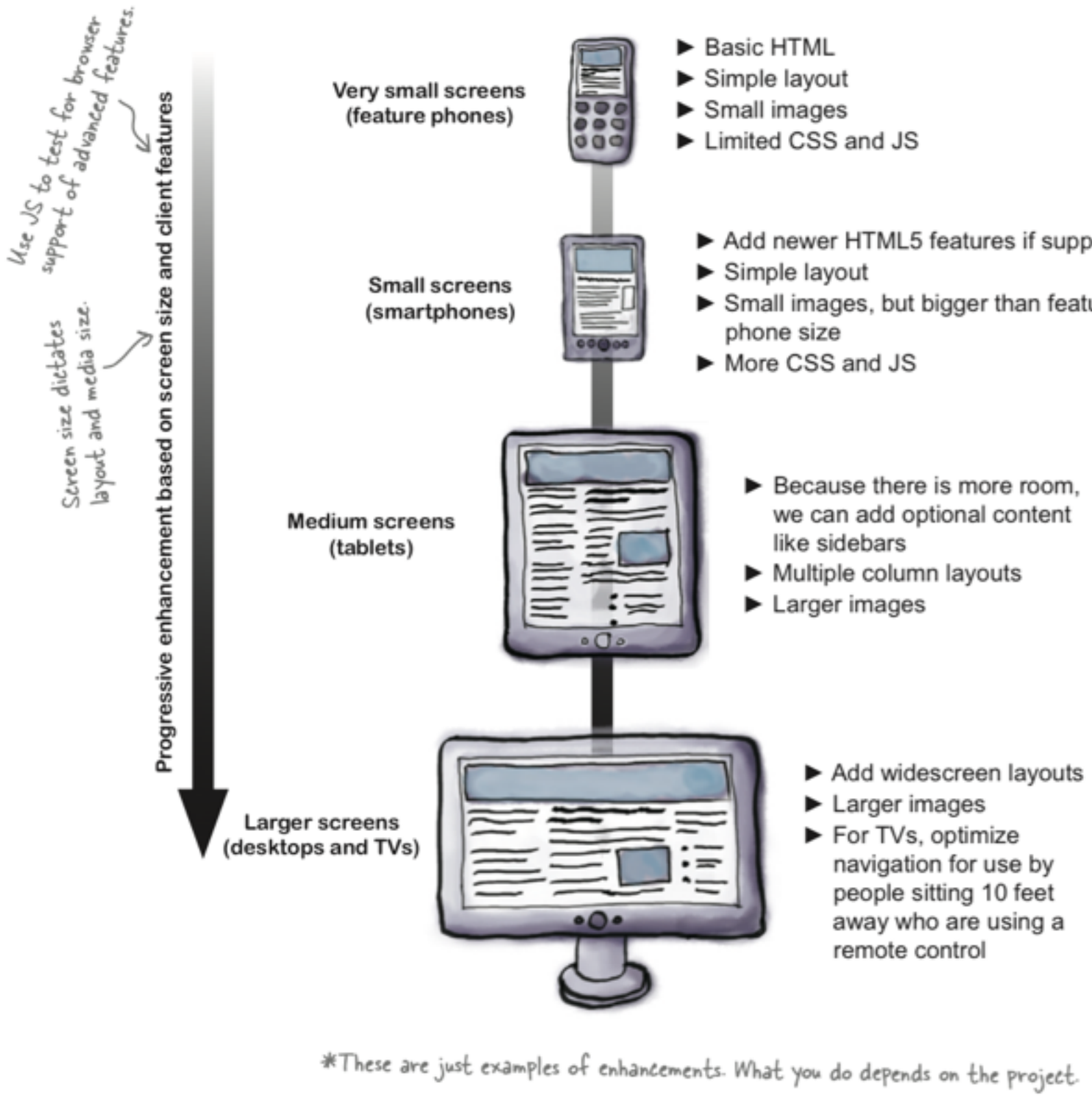
- **cut content**, to reduce word count and defer secondary information to secondary pages; and

- **enlarge interface elements**, to accommodate the "fat finger" problem.

Mobilization Exercise

1. Work through example with instructor
2. Break into teams
    1. Choose a mature product or category
    2. List at least ten major features
    3. Choose the most important one for a mobile user
    4. Sketch a single mobile screen that performs that feature
3. Report to class

Progressive Enhancement



Excerpt From: Lyza Danger Gardner & Jason Grigsby. “Head First Mobile Web.” iBooks. https://itun.es/us/ZENvJ.l

How To Progressively Enhance

- **Step 1: Create an HTML-only layout**
  - No CSS, JS
  - Small, low-res images & scalable vector graphics
  - Leverage HTML5 tag types
    `<article>` instead of `<div class="article">`
  - Leverage the browser's default layout packing behavior for a full-width, single-column layout
  - Use conventions that exist to make styling easier (i.e. a container div) sparingly if at all
  - Make sure the essential functions work without JS

How To
Progressively Enhance

- **Step 2: Use CSS, JS to enhance for mobile**
  - Add CSS & JS to enhance the mobile experience
  - Use modern mobile UI patterns
  - DO NOT change the HTML structure
    (its ok to add classes)
  - DO NOT write media queries for mobile devices

How To Progressively Enhance

- **Step 3: Use CSS, JS to enhance larger screens**
  - Add CSS & JS to enhance the widescreen experience
  - DO NOT change the HTML structure (its ok to add classes)
  - Write media queries for larger devices
    - Use Horizontal layout aesthetics
    - Bring down higher resolution images via CSS
  - Use AJAX to bring in additional content as needed

HTML-only Exercise

- Take the single-function mobile web site wireframe from the previous exercise
- Do a quick sketch in HTML of its basic structure
- Leverage Emmett snippets and Lorem Ipsum gibberish for content
- Don’t worry about images
- Be prepared to show your code to class

Responsive HTML

Use the Viewport

- Set the viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- This allows the page to reflow content to match different screen sizes

- Size content to the viewport

  - Do not use large fixed width elements.

  - Content should not rely on a particular viewport width to render well.

Responsive CSS: Fluid Layouts

Fluid Layouts
Units of measure

- **%** - Percentage of parent container

- **em** - based on font size in *parent* element

- **rem** - based on font size of *root* element

- **vw/vh** - based on viewport size

  vw: hundredths of the viewport width.
  vh: hundredths of the viewport height.
  vmin: hundredths of whichever is smaller, the viewport width or height.
  vmax: hundredths of whichever is larger, the viewport width or height.

http://viewportsizes.com/mine/

Root Em Units

- em - equal to the size of the font that applies to the *parent* of the element in question.
- rem - always equal to the font-size of the HTML root element
- Much easier to use
- Not available in IE8 (http://caniuse.com/#feat=rem)
- Example: http://codepen.io/Auraelius/pen/yyjQJZ

Not everything is fluid

- Widths should always be fluid
- Verticals are not related to viewport width and more related to content and design: “white space” and “rhythm”
- Padding & margin also related to content and design, not viewport width. Sometimes low percentages, sometime rems.

A note of perspective…

- Developers squeeze and stretch layouts all the time

- Laptop users infrequently change window sizes

- Mobile users rarely change window size but sometimes switch between landscape and portrait.

- It’s more important to start with the right page for the viewport size than to transition gracefully from one to another

Responsive CSS: Media Queries

Media Queries

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

- CSS3 **media queries** let us apply CSS selectively to different user environments based on the current value of relevant media features.

- **Media types** (e.g., screen, print, projection) have **media features** (width, color, monochrome, orientation). It's these media features we evaluate in our media queries.

- A **CSS media query is a logical expression**. When it evaluates to TRUE, the enclosed CSS rules are applied.

- There are lots of media features: screen size, screen density, orientation and many others.

  http://pieroxy.net/blog/2012/10/18/media_features_of_the_most_common_devices.html

Syntax

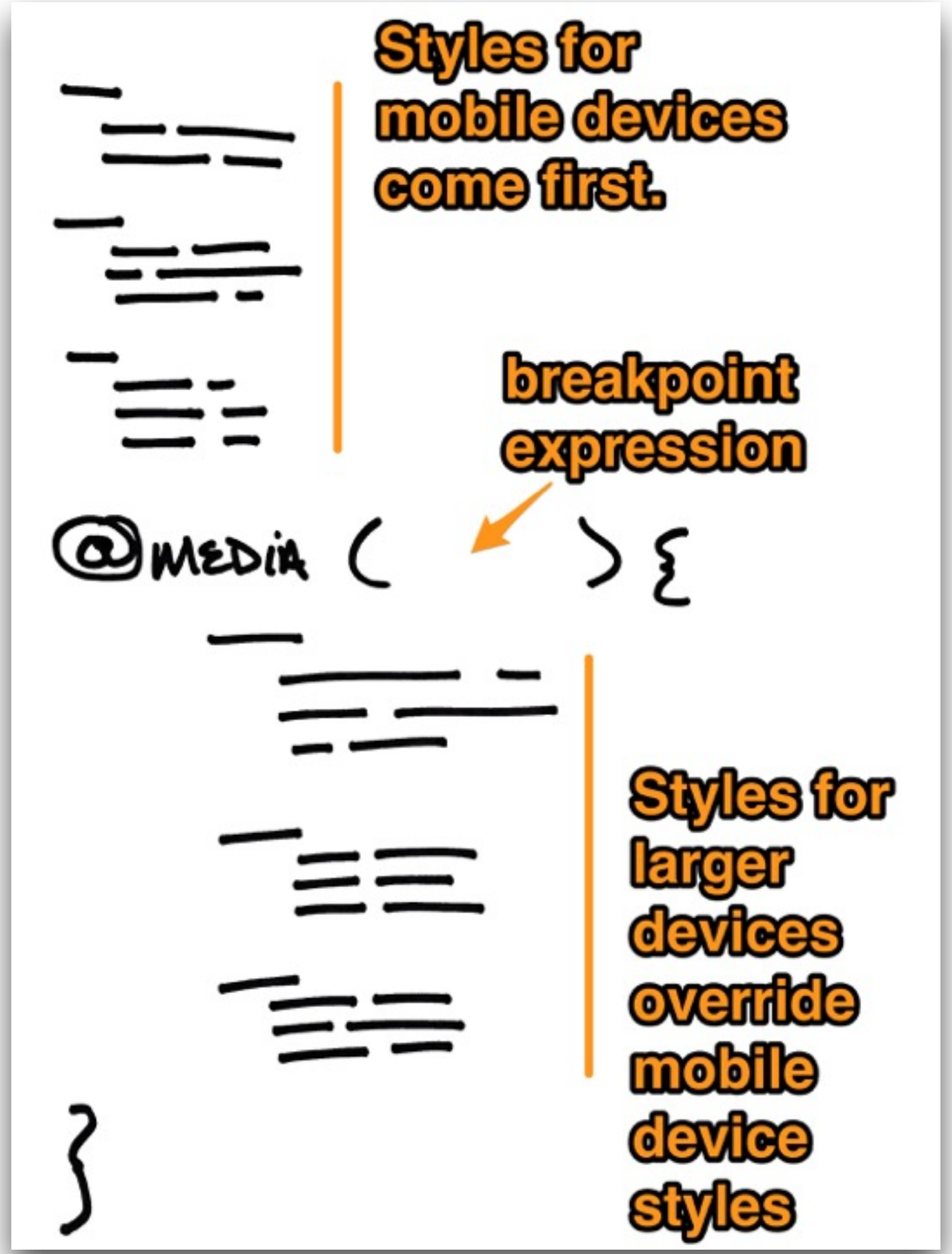
```
<!-- CSS media query on a link element -->
<link rel="stylesheet" media="(max-width: 800px)" href="example.css" />

<!-- CSS media query within a stylesheet -->
<style>
@media (max-width: 600px) {
  .facet_sidebar {
    display: none;
  }
}
</style>
```

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

Mobile-first media queries

- Design mobile styles first.
- Add media query for larger devices.
- Override styles & specific properties as needed.

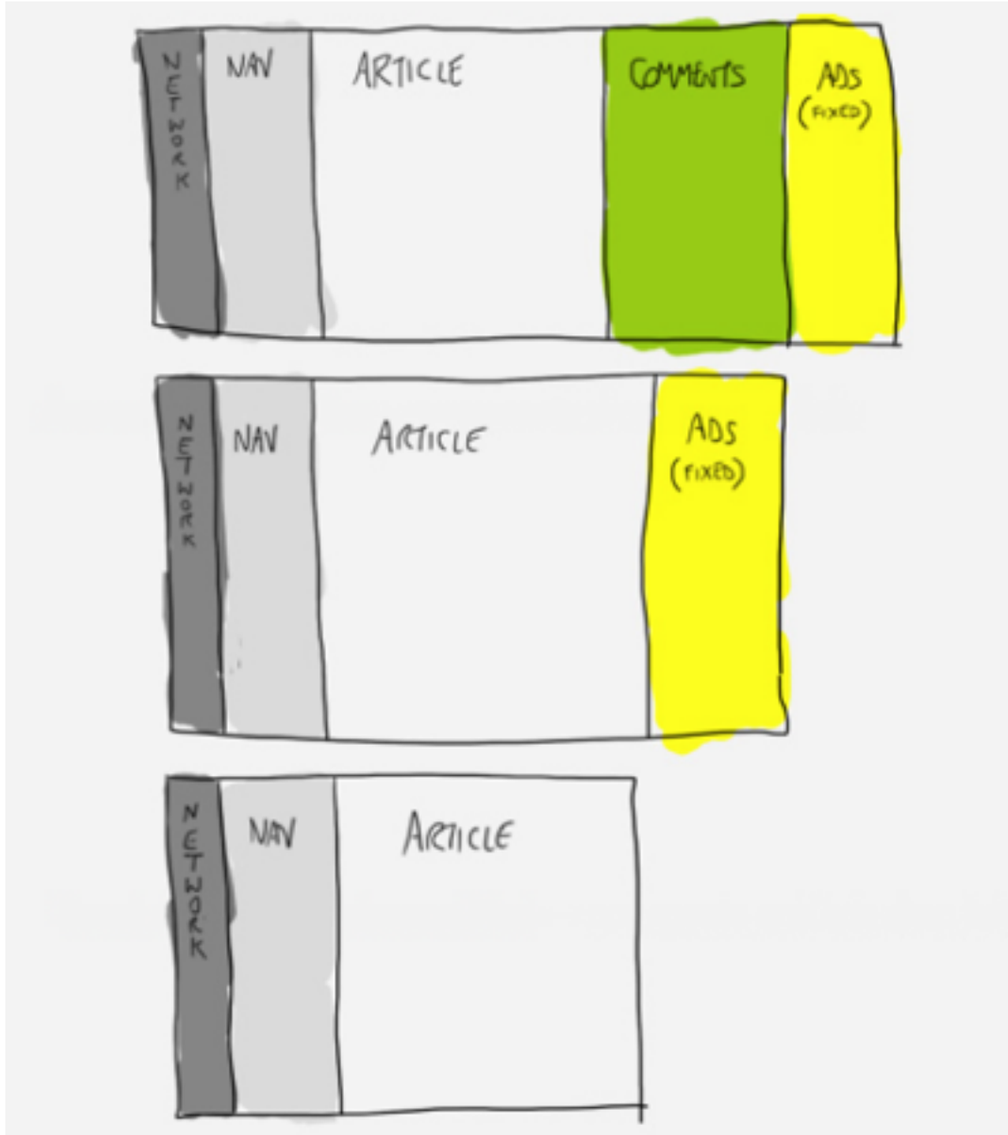


Media query exercise

- Use Codepen.io to create a snippet
- Step 1: Make a box that almost fills the viewport and is blue on a phone but green on a laptop. Use a pixel-based, mobile-first media query.
- Step 2: Fill the box with text. Use a larger font size on mobile than laptop.
- Step 3 (stretch): Use two columns on laptop.

Breakpoints

- Create breakpoints based on content, never on specific devices, products or brands.
- Design for the smallest mobile device first, then progressively enhance the experience as more screen real estate becomes available.
- Keep lines of text to a maximum of around 70 or 80 characters.



https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/how-to-choose-breakpoints
http://www.smashingmagazine.com/2013/03/01/logical-breakpoints-responsive-design/

- Nav example:

http://codepen.io/Auraelius/pen/qdqKBm

Responsive Images

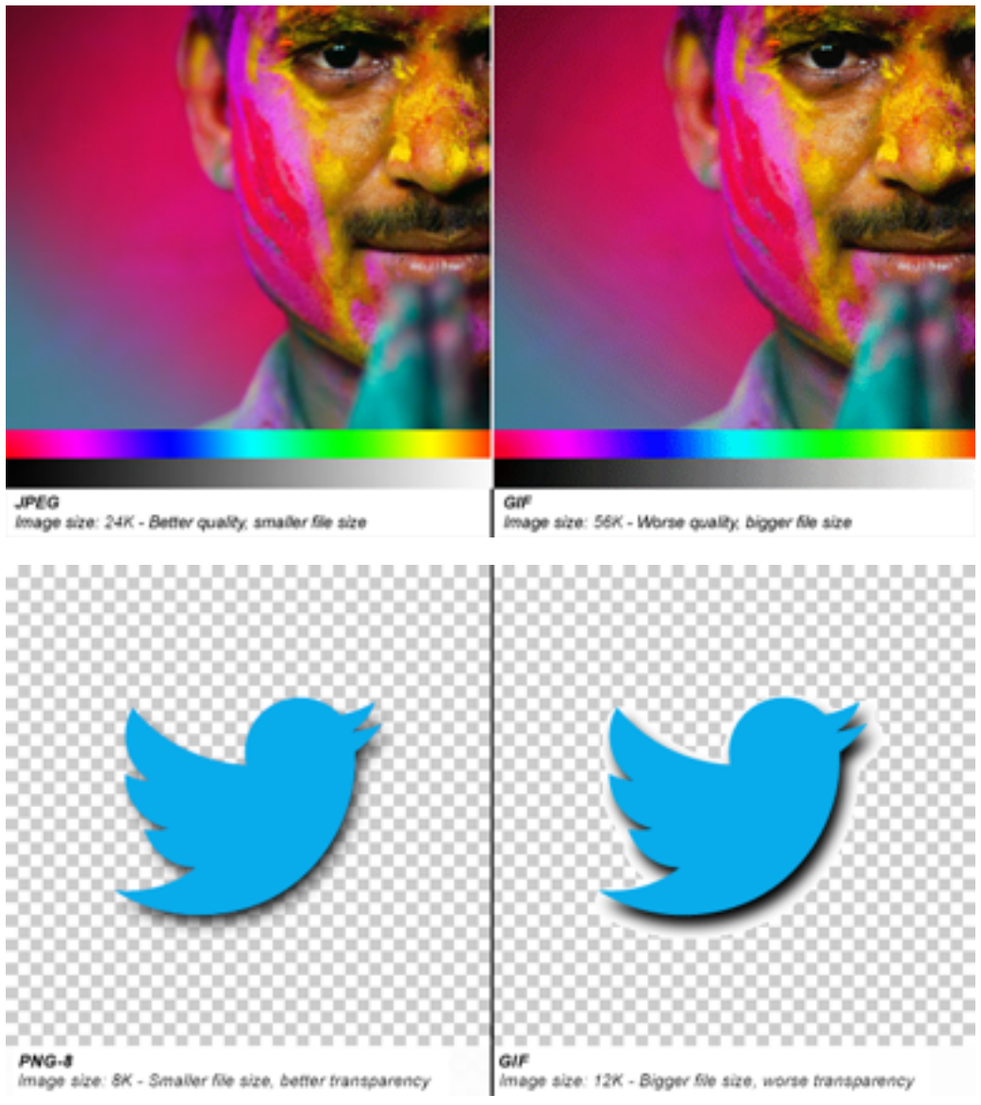
Responsive Images

- Choose image format based on content
- Manage images in HTML/CSS using fluid layout
- Load different image files based on device features

https://smartaddons.s3.amazonaws.com/images/Released-Image/sj-lifemag/new/responsive.png

Start with the right format

- Use JPG for photographic images.
- Use SVG for vector art and solid color graphics such as logos and line art.
- Use PNG-8 rather than GIF as it allows for better transparency, more colors and offers better compression ratios.
- Use GIF for small animations but for longer animations, consider using <video> which provide better image quality and gives the user control over playback.
- Reduce file size



https://developers.google.com/web/fundamentals/media/images/optimize-images-for-performance
http://stackoverflow.com/questions/2336522/png-vs-gif-vs-jpeg-when-best-to-use

Scalable Vector Graphics

- Image format is text (a form of XML) so very small and fast to load
- Drawn by the browser so old browsers have trouble
- Contains default sizes for HTML-only layouts
- Can be included by <img> tag or directly as code.
- See https://thenounproject.com/ for fun examples

- SVG Resizing & Styling example: http://codepen.io/Auraelius/pen/eqcgJ

- Fork the pen, then play with the width of the screen and the styles in the CSS

Look at the code examples at the bottom

- What code determines it's unstyled size? Is this the same as the unsettled <img> version?

- What was added to allow application of styles?

Going further with SVG

- http://tutorials.jenkov.com/svg/index.html

Manage images in CSS

- Set media to 100% width of parent container for fluid layout

```
img, embed, object, video {
  max-width: 100%;
}
```

- Change the background-image or content property in CSS for using media queries

```
.picture {
  content: url(elephant_1620217f.jpg);
}
```

- Serve smaller/cropped/more compressed files to small devices

http://codepen.io/Auraelius/pen/sClka

Aside:

For an interesting perspective on the “Woman laughing alone with her salad” meme, see

http://www.theguardian.com/women-in-leadership/2014/mar/05/woman-laughing-alone-with-salad

Responsive images exercise

- Go to http://codepen.io/Auraelius/pen/zchIq
- Examine the page at different widths
- Answer these questions:

1. What are the media query breakpoints?
2. What tag is used to display the SVG file?
3. What CSS property is used to change the image?
4. Why does the image disappear at some viewport widths?
5. What changes in the HTML to make the caption change?
6. What class & selector controls the changing caption?
7. With the cache disabled, does the page load faster at different widths? Use the network tab to get your numbers.

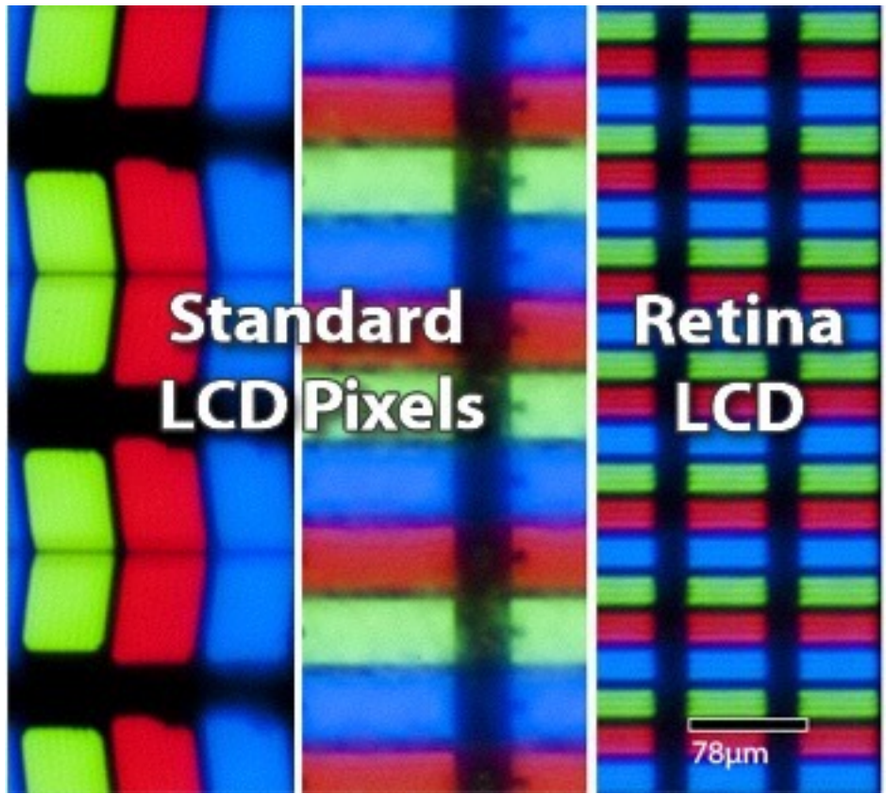
Undiscovered country: Pixel density

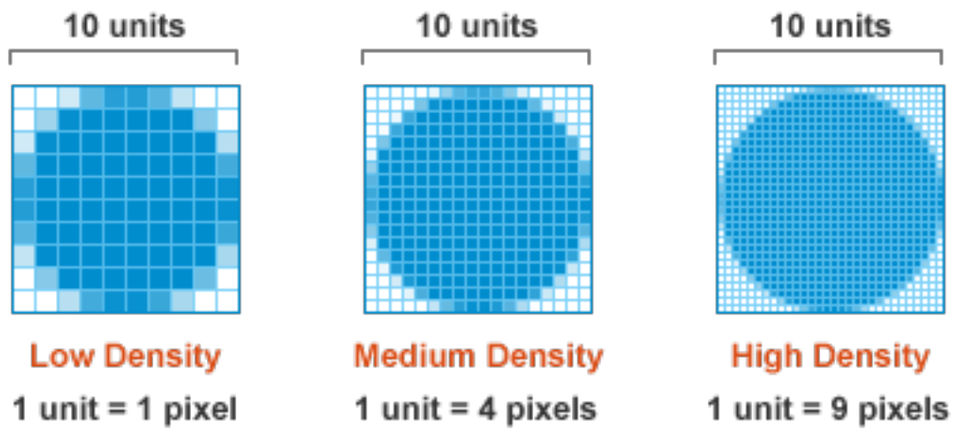
- Change the background-image property in CSS for high DPI displays using media queries with `min-resolution` and `-webkit-min-device-pixel-ratio`.

- Use the `picture` element when you want to specify different images depending on device characteristics.

- Use `srcset` to provide high resolution images in addition to the 1x image in markup.

- Use `srcset` and the x descriptor in the img element to give hints to the browser about the best image to use when choosing from different densities.



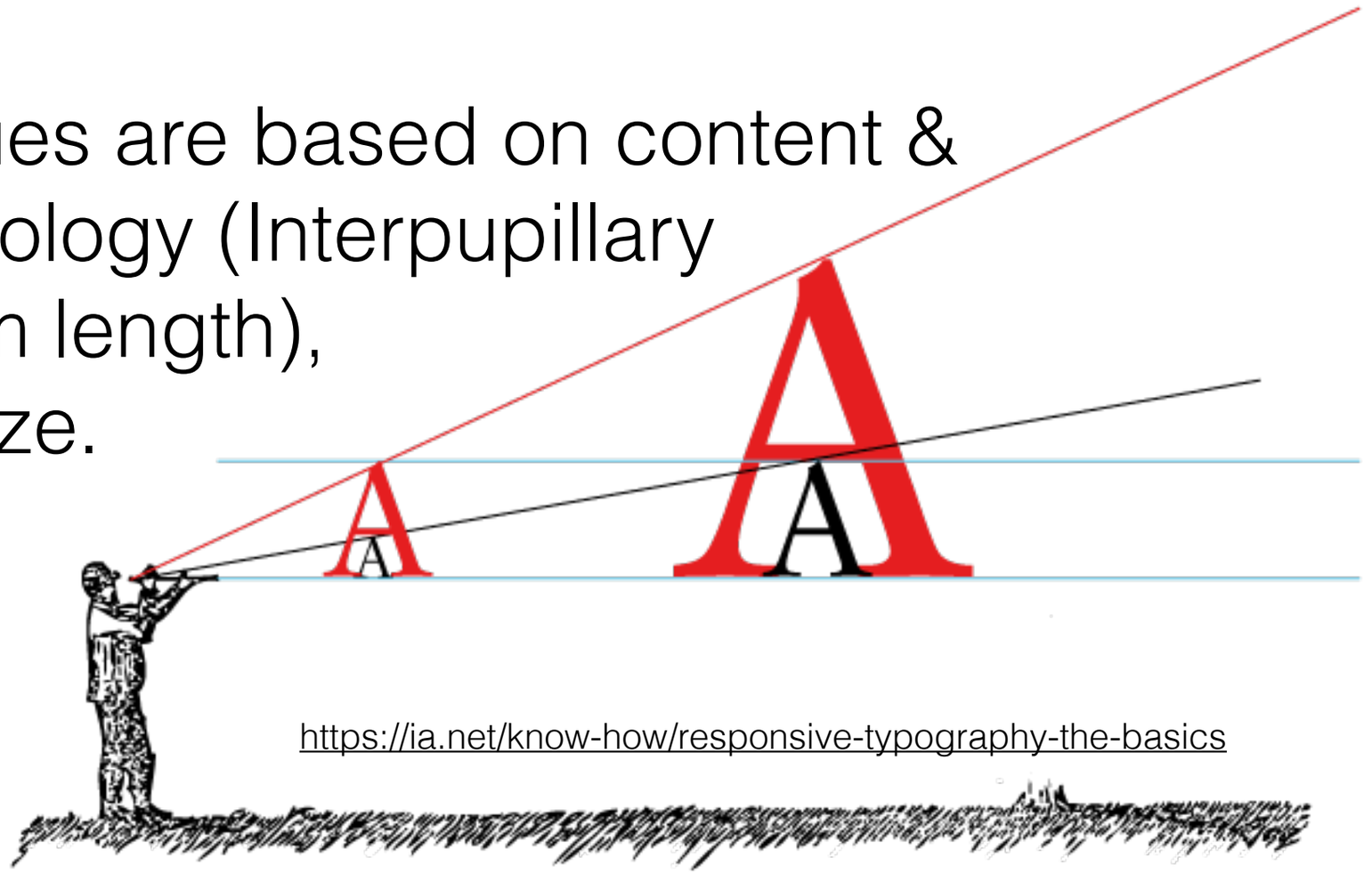


https://developers.google.com/web/fundamentals/media/images/
http://www.techspot.com/images/teaser/retina.jpg
http://www.bbc.co.uk/gel/tablet/tablet-device-considerations/pixel-density

Responsive Typography

Responsive Typography

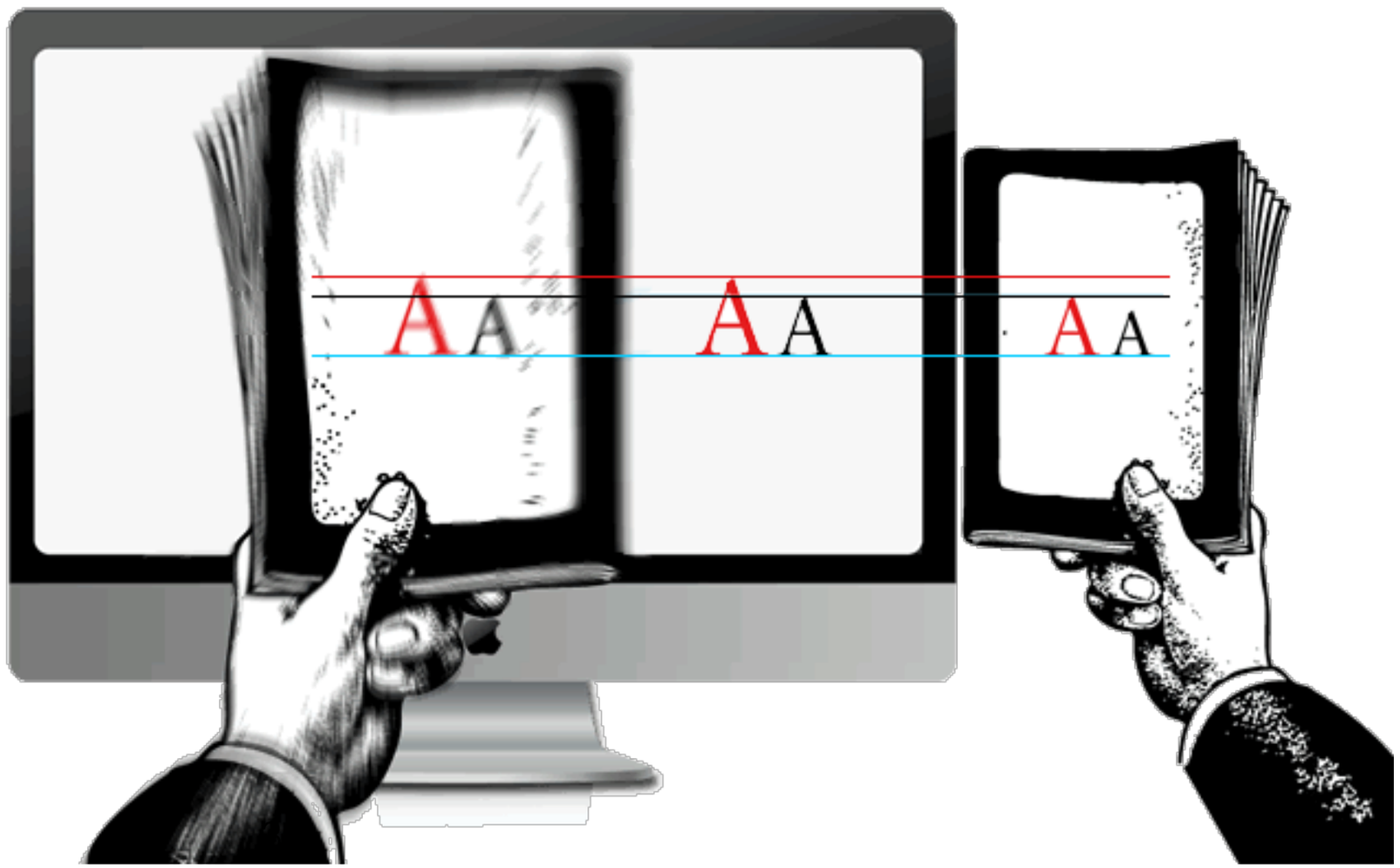
- Text has been around for thousands of years.
- We know how to make text readable.
- Our techniques are based on content & human physiology (Interpupillary distance, arm length), not device size.



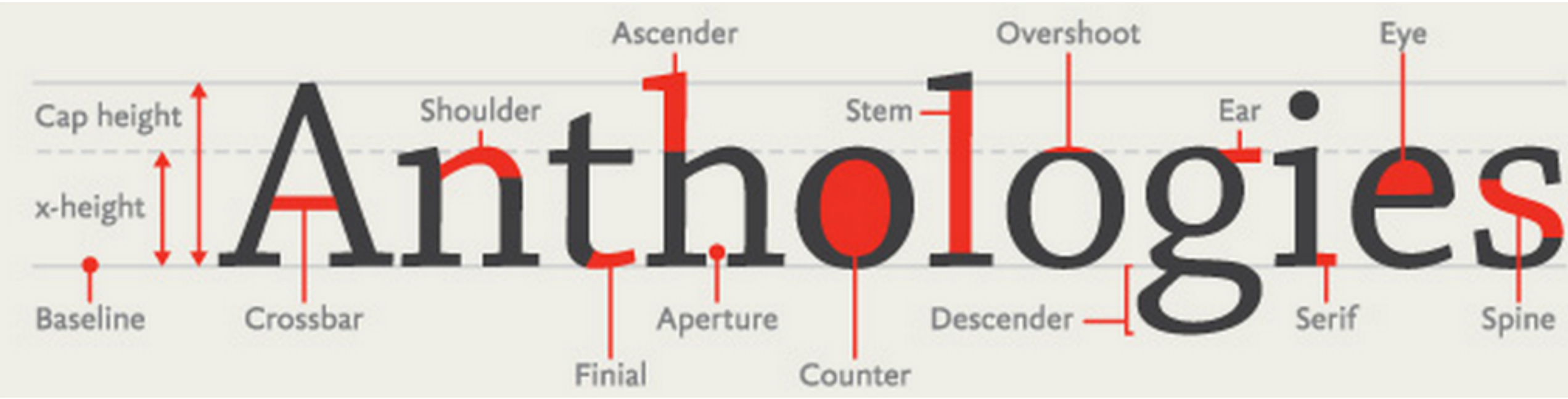
https://ia.net/know-how/responsive-typography-the-basics

Key Aspects

- Typeface
- Column width
- Line height



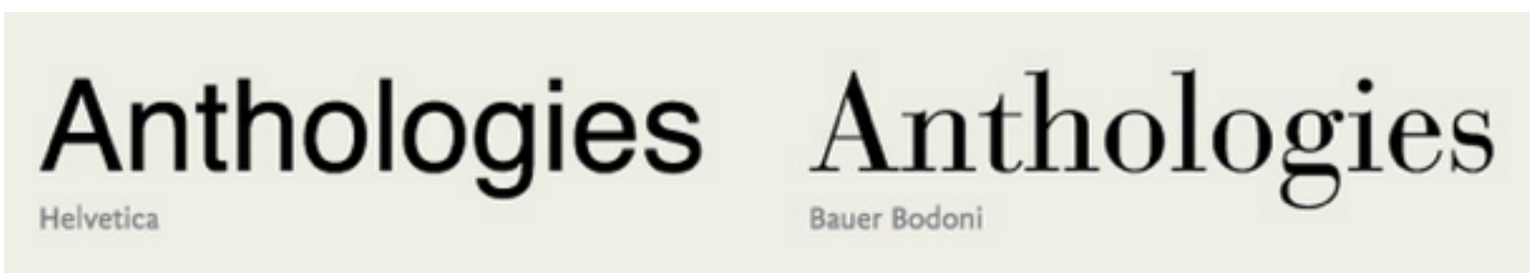
Typeface



- Word shape

- Contrast

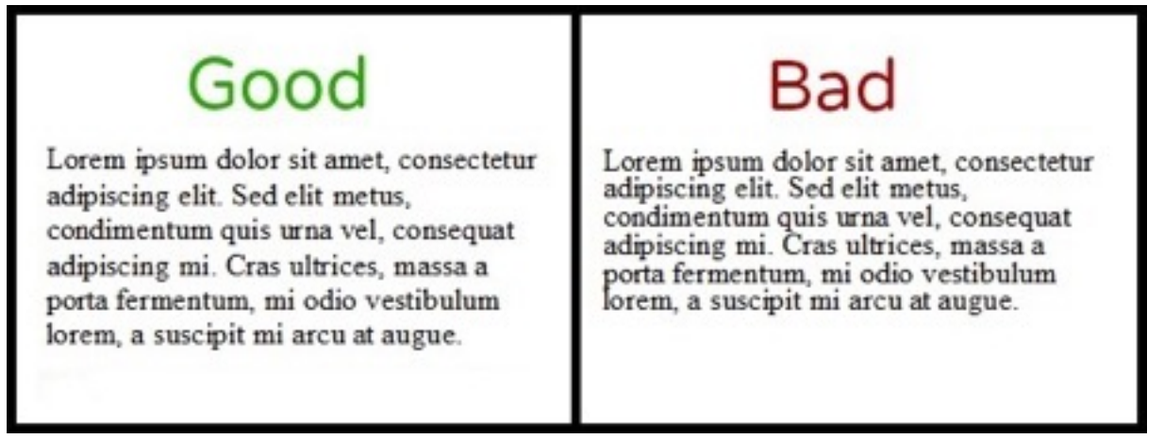
On Web Typography,
Jason Santa Maria,
A Book Apart



Readability

- Line length = 40 to 70 characters per line
  - set column `width` in `rem` units

- Leading - depends on content, font and line length but typically 150% of font height
  - set `line-height` property in `rem` units



http://www.vanseodesign.com/web-design/legible-readable-typography/

http://thecontentauthority.com/blog/wp-content/uploads/2014/02/Content-readability-tip-line-height-or-leading.jpg
http://www.magazinedesigning.com/wp-content/uploads/2013/07/column-width-line-length-character-count.png

Font face

- **System** fonts - always available ‘cos the OS uses them.
- **Common** fonts - frequently present ‘cos everybody (like MS Office) loads them
- **Custom** fonts (downloadable) - best typography but adds download time

```
<style type="text/css" media="screen, print">
  @font-face {
    font-family: "Bitstream Vera Serif Bold";
    src: url("https://mdn.mozillademos.org/files/2468/VeraSeBd.ttf");
  }

  body { font-family: "Bitstream Vera Serif Bold", serif }
</style>
```

Homework:

http://portlandcodeschool.github.io/primer/assignments/04-mobile-first-responsive-design/