ECS 162
Web Programming

Due Thurs 4/11

Exhibit assignment

- Flexbox and media queries (covered today).
- Responsive design which matches the mock-ups (Jamie’s pdfs) as well as possible.
- Harder than it looks; make some progress this weekend.
- We’ll talk about the button on Monday.

Responsive design

- Whatever width the window is,
  - the whole top of the page appears,
  - and it neatly fills the width of the window.
- Nothing ever gets cut off and no extra fill appears on the sides.
- Layout changes smoothly as window is resized, with maybe a jump as we switch from wide to narrower window.

We’re not just showing Dana’s pix

- Your program should never produce views like this
- These are not responsive. Why?

Google fonts

- Google has a collection of fonts that a Web page can download and use. The HTML has to do the download:
  `<link href="https://fonts.googleapis.com/css?family=Montserrat:200,500" rel="stylesheet">`
  font-family: montserrat, sans-serif; font-weight: 200; font-style: normal;
Layout and Flexbox

- Traditional approaches used CSS display modes such as inline-block and float to “trick” the default layout into doing interesting things.
- Tables and later frames were used to arrange chunks of content.
- CSS/Javascript libraries for layout (e.g., Bootstrap) got popular.
- Now (e.g., since 2016) CSS has two built-in layout systems, grid and flex. We’ll work with flex.

Flexbox idea:
Web page is hierarchy of flexbox containers. Within a container, we lay out items (which might be other flex boxes) either vertically or horizontally.
Flexbox containers

- The css display property controls layout.
- By default display is inline or block.
- Another alternative (one of many!):
  display: flex;
- Makes the element a flexbox container. Layout of direct children of the container is handled by flexbox.

Layout of items

- Direct children of a flexbox container are flexbox items. Specify in container how items will be laid out:
  flex-direction: row; /* default */
  or
  flex-direction: column;
- Direction in which items are laid out in the container is the main axis; the perpendicular direction is the cross axis.

Getting text beside pictures

Container vs item commands

- Some flexbox properties are about the html element as a flexbox container:
  flex-direction, justify-content, align-items
- Others are about the html element as a flexbox item:
  flex-basis, flex-grow, flex-shrink
- Since an html element can be both a flexbox container, and an item, keep the groups of properties separated by a space or comment in your css.

Basis, grow, shrink

- Grow is share of excess space item will take up.
- Shrink share of needed space item will contribute if it has to shrink.
- Basis is item’s standard size in the main axis direction; “auto” means however big it has to be to contain it’s content.
- Default:
  grow=0, shrink=1, basis=auto
Grow and shrink are given as shares of excess or needed space, respectively.

- If I get two shares and you get one share, then I get $2/3$ and you get $1/3$.
- If I get one share and you get zero shares, I get it all and you get nothing.

Example: Sticky footer

- We want the footer to sit at the bottom of the page, even if the contents don’t fill up the page.