Grid

A new way to change the layout

Grid

- You can use grid to place your content in columns.
- You need to define a *parent* element and give it *children* elements.

Parent Element

- Step 1:
  - Set display to grid
- Step 2:
  - Set grid-template-columns to number and size of columns
- Step 3:
  - Set justify-content

One column grid

```
div {
    display: grid;
    grid-template-columns:
500px;
}
```

Two column grid

```
div {
    display: grid;
    grid-template-columns: 50% 50%;
}
```

Three column grid

```
div {
    display: grid;
    grid-template-columns: 25% 25% 25%;
}
```

justify-content

- You may want to adjust the default layout of the children with justify-content
- Some of the possible values are:
  - start, end, center, stretch, space-around, space-between, space-evenly
- CSS Tricks: justify-content

Modifying the child elements

- Best practice is to not hardcode the width of the children elements
- Use a fluid measurement to make the most of the parent structure

Positioning the children elements

- The children elements will automatically fall into the next available space
- You can move the element using:
  - grid-column-start
  - grid-column-end

Review

- This is just a very high-level overview of grid.
- Other popular properties include:
  - grid-template-rows, align-items, row-gap, column-gap, etc.
- Using Inspect Element will help you make the most of the options.
- Learn more at A Complete Guide to CSS Grid

University of Michigan
School of Information

Flex

Giving the browser permission to change your layout

Flex / Flexbox

- You can use flex if you want to let the browser resize your elements based on the screen size
- You need to define a *parent* element and give it *children* elements.

Parent Element

- Step 1:
  - Set display to flex
- Step 2:
  - Set flex-wrap to wrap or nowrap
- Step 3:
  - Set flex-direction to row or column
- Step 4:
  - Set the justify-items and/or align content

Default flex

```
div {
  display: flex;
 }
```

Using wrap

```
div {
  display: flex;
  flex-wrap: wrap;
}
```

Changing the layout direction

```
div {
  display: flex;
  flex-direction: column;

}
```

Adjusting the spacing

```
div {
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

justify-content

- You may want to adjust the default layout of the children with justify-content
- Some of the possible values are:
  - flex-start, flex-end, center,, space-around, space-between, space-evenly
- CSS Tricks: justify-content

But…..

- You use justify-content when the direction is row.
- If you are using direction column you will want to use align-content instead.
- Some of the possible values are:
  - start, end, center, stretch, space-around, space-between, space-evenly
- CSS Tricks: align-content

Review

- This is just a very high-level overview of flex.
- There are many more options including
  - flex-flow, row-gap, align-items, shrink, grow, order
- Using Inspect Element will help you make the most of the options.
- Learn more at A Complete Guide to Flexbox | CSS-Tricks

University of Michigan
School of Information

Styling Links and Lists

Anchor Links

- Links can take on all of the usual styles as well as *text-decoration*

This is a link

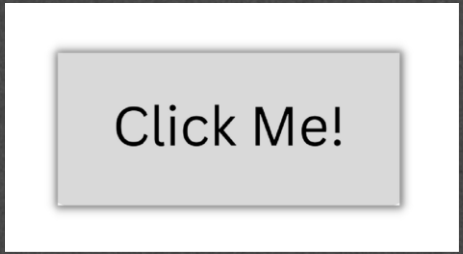
This is a link

```
a {
    display: block;
    font-weight: bold;
    color: #ffffff;
    background-color:
#0006CC;
    width: 200px;
    text-align: center;
    padding: 4px;
    text-decoration: none;
}
```

"Buttons"

- Many designers try to make their links look like buttons.
- Be semantic, if you want a button use the <button> element instead.

```
<button>Click Me!</button>
```



States

- Some links are blue, some are purple, etc. Why???
  - a:link: a normal, unvisited link
  - a:visited has been visited
  - a:hover activated by mouse
  - a:focus activated with the keyboard
  - a:active is being clicked

Precedence of Rules

- a:hover MUST come after a:link
- a:visited and a:active MUST come after a:hover

Styling Lists

- Number of properties beyond font, margin, etc.
  - list-style-type
  - list-style-image
  - list-style-position
  - list-style

list-style-type

- list-style-type
  - ordered lists
    - lower-roman, upper-roman, decimal, decimal-leading-zero, upper-alpha, lower-alpha, hebrew, armenian, …..

1. Knight Rider
2. A-Team

```
ul {
    list-style-type: upper-alpha;
}
```

A. Knight Rider
B. A-Team

List styles

- list-style-type
  - unordered lists
    - Override the default marker with circles, discs, or squares
- list-style-image
  - Use a custom image instead of traditional marker

```
ul {
    list-style-image: url('icon.gif');
}
```

Review

- At this point you have learned how to write rules for the *tags*.
- Embrace the many tools that are available to help you design your site.
- http://chrispederick.com/work/web-developer/
- http://css3generator.com/
- Do web search for "Developer Tools"

University of Michigan
School of Information

Advanced Selectors

Styling Specific Objects

- We have focused on *type* selectors.
- What if you don't want to style *all* of the links, just some? Or just some of the lists?
- CSS gives you options

CSS Selectors

- Some selectors follow the DOM
- Descendant selectors (nav a)
  - Style all of the anchor links inside a nav tag
- Child selectors (nav > a)
  - more constraining The anchor elements must be a child of the nav, no intermediate tags, e.g. paragraph
- Adjacent sibling (h1 + o)
  - elements must be at same level and follow each other

id Selectors

- # id selector
  - Used to identify a single element in the DOM.
  - Was used extensively for , , etc.
  - There is a small movement to move the use of id OUT of CSS

```
<img src="logo.jpg"
id="mainLogo" alt="logo"/>
```

```
#mainLogo {
    border: 5px solid #0006CC;
    margin: 0 auto;
}
```

class Selector

- . class selector
  - Used to identify an element in the DOM that is part of a special class of items
  - Think of thumbnail images, all of the links that are in the navigation, your social media images, etc….

```
.thumb {
   border: 1px solid #0006CC;
   width: 20%;
}
```

```
<img src="cat.jpg" class="thumb" alt="Joe"/>
<img src="dog.jpg" class="thumb" alt="Bacon"/>
<img src="bird.jpg" class="thumb" alt="Tweety"/>
```

classes vs. ids

- Syntax is “.” and “#”
- classes can be used multiple times
- id should be unique
- Think of images and navigation bars
  - Format numerous (but not all) images the same way
  - Visually signify the current page

Narrowing the Scope

- As you get more advanced pages, you will want to narrow the scope of the of action
- p.main → paragraphs using main class
- header img.special → paragraphs inside header that use special class

Expanding the scope

- You can combine elements with a comma
  - p, h1, #main, .special{…rules to apply to all of them…}
- Review : What happens when there are multiple rules for the same selector?
  - When there are conflicts, use the one processed most recently
  - UNLESS a rule has !important

Whew!!!

- We have actually covered a lot in this short video
- Know that each of these ideas can merge. One element can have many classes and ids associated with it

```
<li class="special early dark" id="main"></li>
```

- Browser "starts at the top" and applies each rule, sometimes overriding earlier rules.

The Good News

- You can use style sheets from others to style your code, just by adding class!!
- You can override style sheets from others just by rewriting the class, or making your own version of it and linking it last.

Review

- Type selectors can be combined to narrow the scope of where rules are applied
- An id is used to specify a specific element in a page
- Classes can be used to associate elements that should treated in a similar manner

University of Michigan
School of Information

Attribute Selectors

Attribute Selectors

- Universal
  - * applies styling to every element on the page
  - Ackk!! Try this!
- Attribute Selectors
  - a[href='info.html']
- PseudoClasses
- Pseudo Elements

Attribute selectors

- You may want to search the DOM for certain elements that have an attribute you are looking for
  - All the images that use gif files…..
  - All of the images that have empty alt text….
  - All of the links that go to government sites….

Using Operators

- Operators can be used to find those attribute values you are looking for
  - ^ : match the beginning exactly
    - a [href^='http://umich']
  - $: match the end exactly
    - img[src$ = '.png'] → apply to .png images
  - * : wildcard
    - a [href*='umich']

Review

- Type selectors can be combined to narrow the scope of where rules are applied
- An id is used to specify a specific element in a page
- Classes can be used to associate elements that should treated in a similar manner

University of Michigan
School of Information

Browser Capabilities

Designing for Consistent Appearance

Browsers Differ

- Even though browsers are moving to a consistent implementation of HTML, they differ in display and adherence.
- It is your responsibility to make sure your page works for a wide audience.

Handling Stylistic Differences

- "Easiest" way to eliminate browser differences is to use a default style sheet
- Default style sheets reset all of the values for the page
- Will make your page look worse!

Handling Unsupported Properties

- Not all browsers support all HTML5 tags
- Not all browsers support all CSS3 properties
- Browser prefixes (or vendor prefixes) provide a quick fix for handling unsupported CSS3 options.

Browser Prefixes

- -webkit-: Android, Chrome, iOS, Safari
- -moz-: Firefox
- -ms-: Internet Explorer
- -o-: Opera

Often Unsupported Properties

- column-count
- border-radius
- gradient
- Sites such as http://caniuse.com/ will tell you when you need to use prefixes

Automated Ways to include Prefixes

- For now, add the prefixes by hand
- There are ways to automate the addition of prefixes
  - Editor add-ons (You have most of the control)
  - Use outside programs to dynamically add appropriate prefix based on browser

Review

- Default style sheets remove stylistic differences
  - Should default style sheet be internal or external?
  - Where should it go in relation to other style sheets?
- Browser prefixes can help remove some differences caused by unsupported options
  - Shouldn't be overused

University of Michigan
School of Information

Code Together

Background Images and Opacity

Background Images

- If an image is purely decorative you may want to add it as a background-image rather than using an image tag.
- The syntax is:

```
background-image: url('file_path');
```

Complementary Properties

```
/* Set a background color */
   background-color: black;

/* Set a specified height */
   height: 500px;

  /* Center the image */
   background-position: center;

  /* Resize the background image
    background-size: cover;
```

Opacity

- The opacity property specifies the transparency of an element.
  - 0 is completely transparent
  - .5 is the halfway mark
  - 1 is the default opacity.
- When applied to an element it changes *everything*, not just the background-image.

University of Michigan
School of Information

Designing for Accessibility

POUR

Overview

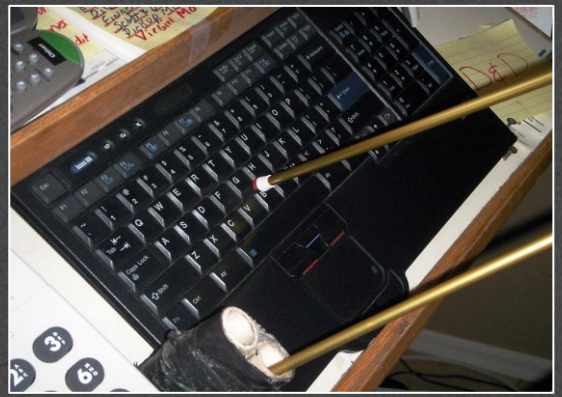
- The content of your page should be in the HTML.
- It is tempting to add content via colors, images, etc.
- Follow the POUR guidelines
  - Perceivable, Operable, Understandable, Robust

Perceivable

- Provide text alternatives for images
- Provide captions and transcripts for video and audio
- Use correct semantic markup so content can be presented in different ways
- Make it easier for users to see content by using good color contrast

Operable

- All functionality available from the keyboard!
- Users have control over timing and limits
- Do not cause seizures (don't flash content)
- Provide ways to help users navigate, find content, and determine where they are




AbleNet
OPTIMA

Understandable

- Economical and plain use of language
- Text supplemented with illustrations, videos, and other formats where appropriate (i.e., use good Universal Design)
- Navigation, information structure are discernable and consistent
- Make pages operate in predictable ways
- Help users avoid and correct mistakes

Robust

- Is your site functional across various technologies (smart phone, screen reader, laptop, pensticks, etc..)?
- Syntax errors that don't affect visual presentation may hamper assistive technology and accessibility tools
- Adhering to W3C standards ensures future compatibility
- Validate your code at validator.w3c.org and wave.webaim.org

Review

- Accessibility starts with proper HTML tags
- Styling can actually make it HARDER for some people to access the information
- Get into the early habit of utilizing accessibility tools
- “Cool” new style should not be at the cost of accessibility

University of Michigan
School of Information

Homework Two

Advanced Selectors and Display

Objective

- Build on your earlier work to use more advanced styling
- Include "Skip to main content" links to improve accessibility.

Getting Started

- You must complete the first Peer Graded Assignment before you can begin this one.
- You can alter your previous styling choices but I assume those changes are complete.

Before and After

- End of Week One
- End of Week Two

Primary Elements Changed

- li
- nav
- ul
- images (using advanced selectors)
- divs (using classes)

Primary Properties Changed

- display (inline-block, grid, flex)
- width
- list-style-type
- grid-template-columns
- justify-content and align-items
- row-gap
- flex-wrap

Peer grading

- Grades will be based on level of completion
- Some aesthetics will come into play this time. It is important that things are not "squished" together
- Proper standards do apply
- You can specify your preferred screen size for grading.

University of Michigan
School of Information