CS 135I Winter 2024

# Week 2 Classwork

## Overview

We continue looking at interesting CSS3 layout tools this week. I’d recommend using the File menu to either download a Word version of this sheet or to create a copy on your own Google drive. Fill out the notes as we go along in class, and complete the activity so that you will be able to upload your proof of completion to Moodle at the end of class.

## CSS Grid Tools

Concept: Flexboxes are great for organizing things in one direction (either rows or columns), but when you have specific needs for organizing in both rows AND columns, CSS grids are the way to go

Creating CSS Grids involves:

1. Creating HTML so that the items you wish to appear in rows and columns are direct children of a parent element.
2. Adding \_\_\_\_\_\_\_\_\_\_\_\_\_ to the styles for the \_\_\_\_\_\_\_\_\_\_\_\_ in order to make a grid-container.
3. Determining the number of columns in your grid by adding the \_\_\_\_\_\_\_\_\_\_\_\_\_ property to the styles for the \_\_\_\_\_\_\_\_\_\_\_\_ .
4. Determining the number of rows in your grid by adding the \_\_\_\_\_\_\_\_\_\_\_\_\_ property to the styles for the \_\_\_\_\_\_\_\_\_\_\_\_ .

Sizing rows and columns requires developers to specify a unit. Developers can indicate the size of a row or column using any of the following units:

* Absolute length (measured in px, em, pt, etc. - any unit supported by CSS)
* Percentages
* \_\_\_\_\_\_\_\_\_\_\_\_ (fr) in order to allow the browser to complete the calculations necessary to divide the space

Some common flexbox-related terms and properties:  
(If you are not in class, the [*Complete Guide to CSS Grid*](https://css-tricks.com/snippets/css/complete-guide-grid/) will be handy in completing the table)

| Property Name | Sample Values | Description | Flex-container =or= flex-item =or both? |
| --- | --- | --- | --- |
| grid-template-areas |  |  |  |
| grid-row-gap/ grid-colulmn-gap/ grid-gap |  |  |  |
| justify-items |  |  |  |
| align-items |  |  |  |
| grid-auto-rows/ grid-auto-columns/  grid-auto-flow |  |  |  |
| grid-column-start/ grid-column-end/ grid-column |  |  |  |
| grid-row-start/ grid-row-end/ grid-row |  |  |  |
| grid-area |  |  |  |
| justify-self |  |  |  |
| align-self |  |  |  |

Building Grid Fluency

* Play [Grid Attack](https://codingfantasy.com/games/css-grid-attack) - defeat the monsters, rescue the brother, and boost your understanding of CSS grids
  + I would recommend the easy level as you get started. You’ll get more narration about the properties and if you get stuck in our limited class time you can use the show answer link to move forward. Be sure to look at the answer and go back to review any properties that were a bit mysterious the first time around!
  + You’ll need to make an account (free - just need to trade them an email address) to get past the first few levels. They do send a confirmation code, so it will need to be an address you can check.
* Paste a screenshot of your successful encounter with the final level here:  
  (Displaying the “*Congratulations, you defeated the last boss!”* message)

**(EXTRA CREDIT FOR WINTER 2024)** Applying Grid Concepts to a Webpage

1. Download the classwork02.zip file from Moodle
2. Extract the contents, and examine both the existing *index.html* file and the *grid.css* file.
3. Update the *index.html* file so that:
   1. The spring catalog <header></header> and all the product <article></article> tags are contained inside of a container element of your choice (div, section, etc.)
4. Update the *grid.css* file using grid-related properties so that:
   1. The products are arranged into a grid that looks like the following:

| *catheader* | | | | |
| --- | --- | --- | --- | --- |
|  | *feature* |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Note each of the following:
     1. The grid should have 4 rows and 5 columns
     2. The first grid track should be a grid area named *catheader*
     3. There should also be a grid area that fills the last 3 cells of the second column named *feature*
     4. The second column should be twice as wide as the other columns
     5. The spring catalog header should fill the entire *catheader* grid area
     6. The product <div></div> with an id of *feature* should display in the *feature* grid area
     7. Ensure that each product division is horizontally and vertically centered inside of its grid cell.
     8. Define the item <div> tags to also be a grid. Each <div> tag grid should look like:

| Product Image (picture should appear horizontally and vertically centered). The image should fill the space allotted to the div, cropping any overage as necessary. | | |
| --- | --- | --- |
| Product Name | *Empty cell* | Product Price |

* 1. Update the media query for smaller browsers so that
     1. The catalog <div></div> items previously organized in the grid show in a simple, vertical display
        1. Each catalog <div> should span the full width of the browser
        2. Each catalog <div> should continue to act as a grid container, with the layout from step *viii* in place.
     2. The div with an id of *feature* should be the first item to show after the *Spring Catalog* heading