

# CSS Can be hard

- Not a way of thinking you may be used to
- "simple" concepts turn out to be hard
  - centering
- efforts often break things that were working
  - fixed widths

# Why and What is Flexbox?

Base CSS is all based on how items align in flow

- no grouping outside of containers with flow
- everything based on the needs of content

Flexbox tries to fill the container with content

- in ONE dimension

# Weird Flex, but....

Core concept: Apply `display: flex;` to parent container

Flexbox will then distribute the space for and around the children

Additional changes are done either to

- the container
  - affects all children
  - or space between them
- to the children
  - affects that child

# Guides to Flexbox

Remember the difference between

- properties on parent
- properties on children

A handy game-tutorial

- **<https://flexboxfroggy.com/>**

CSS Tricks is always a great source:

- **<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>**

# Common Flexbox properties

- `flex-direction` (column or row)
- `justify-content` space along main axis
- `align-items` space along cross axis

# CSS Grids

Control the placement of children within a container

- in TWO dimensions

Grids mimic the old table-based layouts

- without their pain
- because layout is (mostly) separate from structure

Children can be told to span multiple "cells" of the grid

Grids put the emphasis on the layout over the content

# How to Grid

Set parent container to `display: grid;`

- define template columns or rows
- can define areas

# Guides to CSS Grids

A game-tutorial

- **<https://cssgridgarden.com/>**

CSS Tricks

- **<https://css-tricks.com/snippets/css/complete-guide-grid/>**

Debugging with Chrome

- **<https://developers.google.com/codelabs/devtools-debug-css-grid#0>**



# Grid areas (Other options later)

- Parent element:
  - `grid-template-areas`
    - double quoted strings of space separated labels
      - No commas!
    - One string per row, label is column
  - `grid-template-rows`, `grid-template-columns`
    - defines sizes of cells
    - `%` of available
    - `auto` matches needed space for content
    - `1fr` (or `2fr`, etc), divvy up remaining

# Grid areas, Child Elements

- `grid-area`
  - Give label (no quotes)

# Summary - Flow

Big common concepts in layout

- inline (flow of text)
- block (sections to organize)

Controlled by `display` property

Inline has limited sizing options (because text)

Block has width AND breaks flow

`inline-block` - has inline flow, but block-like sizing

# Summary - Flexbox

Layout beyond inline and block

Organizes *child* elements in one dimension

- distributes space
- distributes space between/around

set by `display: flex;` (NOT `flexbox`)

- other properties on parent and child

# Summary - CSS Grid

Another option for layout beyond inline/block

Organizes *child* elements in two dimensions

- has "cells" to distribute space

set by `display: grid;`

- other properties on parent and child
- set sizes for columns and/or rows

Can label cells for content

- content can span many cells

Other options for assigning cells covered later