#### **Inline vs Block**

Elements are naturally either:

- inline
  - they take up size based on content
  - CSS resizing highly limited
  - do not break the "flow" of text
- block
  - they will shift to fill width of container
  - height as needed by content
  - CSS resizing fully available
  - break flow before and after

Rules set by display property

### Notes about inline elements

- Do not break flow
  - means some sizing properties don't do anything

### **Notes about Block elements**

Take up full-width of container by default

• AND break flow

Breaking flow means changing the size alone won't stop it

#### Notes about inline block elements

```
display: inline-block;
```

- Does not break flow
- Does allow for resizing

If you are changing display, it will tend to be to inline-block or one of the layout options

• Don't swap inline to block or vice-versa

# Notes about floating

```
float: left; (etc)
```

Used to have inline elements flow around it

• e.g. a paragraph of text wrapping around a small image

#### DO NOT USE TO FAKE LAYOUT

was a common fix before flexbox/grids

#### **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

• <a href="https://flexboxfroggy.com/">https://flexboxfroggy.com/</a>

CSS Tricks is always a great source:

• <a href="https://css-tricks.com/snippets/css/a-guide-to-flexbox/">https://css-tricks.com/snippets/css/a-guide-to-flexbox/</a>

### **CSS Grids**

CSS Grids allow you to control the placement of children within a container

• in TWO dimensions

Grids mimic the old table-based layouts, but 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

• <a href="https://cssgridgarden.com/">https://cssgridgarden.com/</a>

**CSS Tricks** 

• <a href="https://css-tricks.com/snippets/css/complete-guide-grid/">https://css-tricks.com/snippets/css/complete-guide-grid/</a>

Debugging with Chrome

• <a href="https://developers.google.com/codelabs/devtools-debug-css-grid#0">https://developers.google.com/codelabs/devtools-debug-css-grid#0</a>

# **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