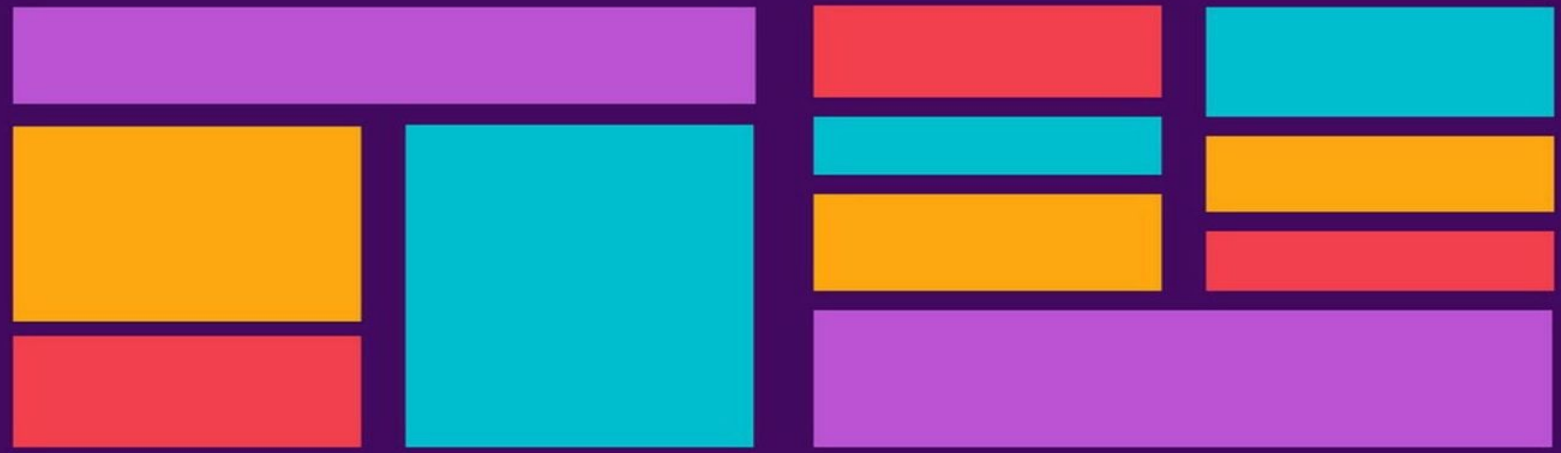
Facilitation Guide

Here are some quick tips for facilitating engaging classes:

- Skip to Slide 2 and use presenter view. You can still view your notes and engage with your audience!
- The notes contain helpful lines of inquiry to lead your class with
- Additional resources(videos, articles, exercises) for this lesson are in the facilitation guide
- Review your TKH instructional/ pedagogy guide for further help

You should…

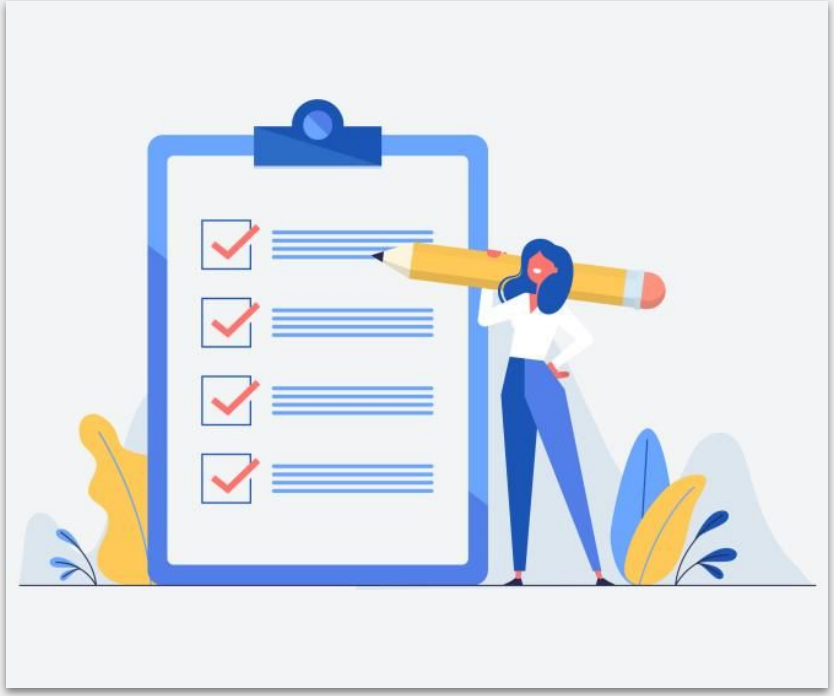
1. Introduce yourself and share some of your experience
2. Ask questions and debrief
3. Provide opportunities for learners to collaborate
4. Collect survey data(when possible)
5. Thank your participants for their time and energy
6. Share feedback with your team
7. Have **FUN!**



CSS Layouts with Flex

Learning Objectives

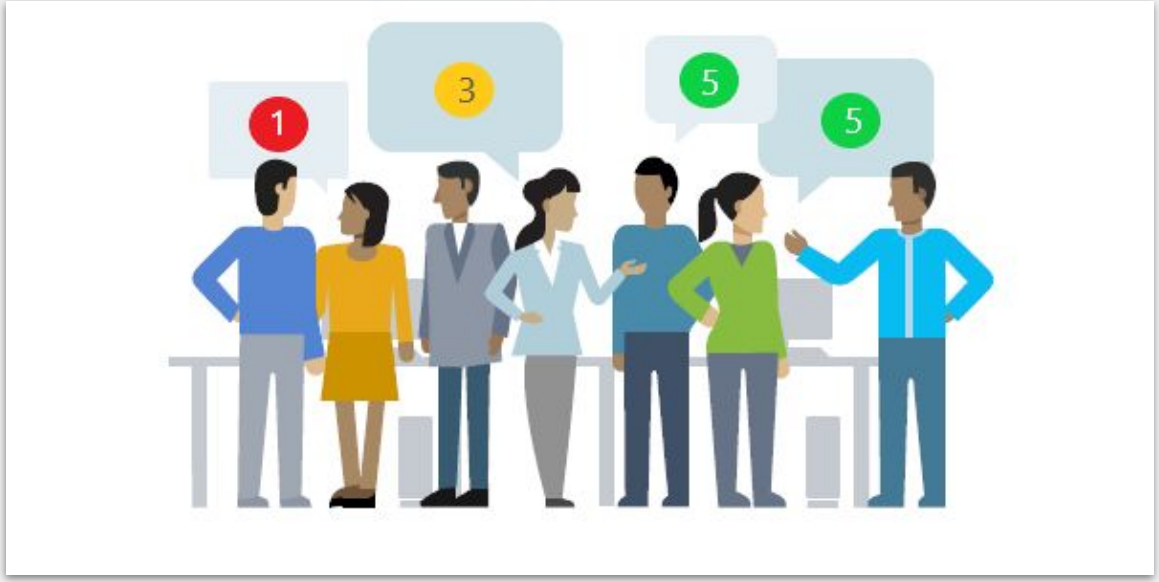
1. Develop familiarity with Cascading Style Sheets
2. Understand the importance of Flex in web design
3. Practice using CSS Flex to position elements on a page



Fist-to-Five

Raise your hand with the amount of fingers that correspond to your response

- 1 = Never heard of CSS Flex before today
- 3 = Have played Flex Froggy or Zombies
- 5 = Made responsive layouts with Flex



Warm-up

Share responses to the following amongst other members

- How do you currently position elements on a webpage?
- Have you ever struggled with positioning elements on a webpage?

The Limits of HTML

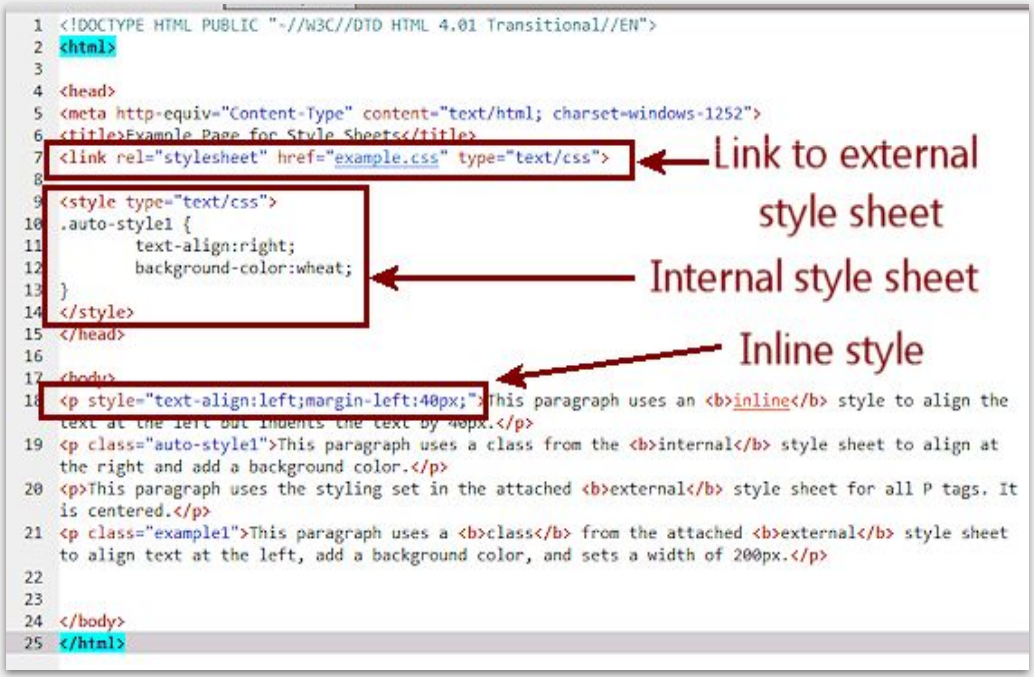
While HTML was a huge step forward, alone it only created static web pages with very limited style and no interaction

CSS Brings Style to the Web

CSS was created to give webpages new structure, organization, and best of all, **STYLE!**

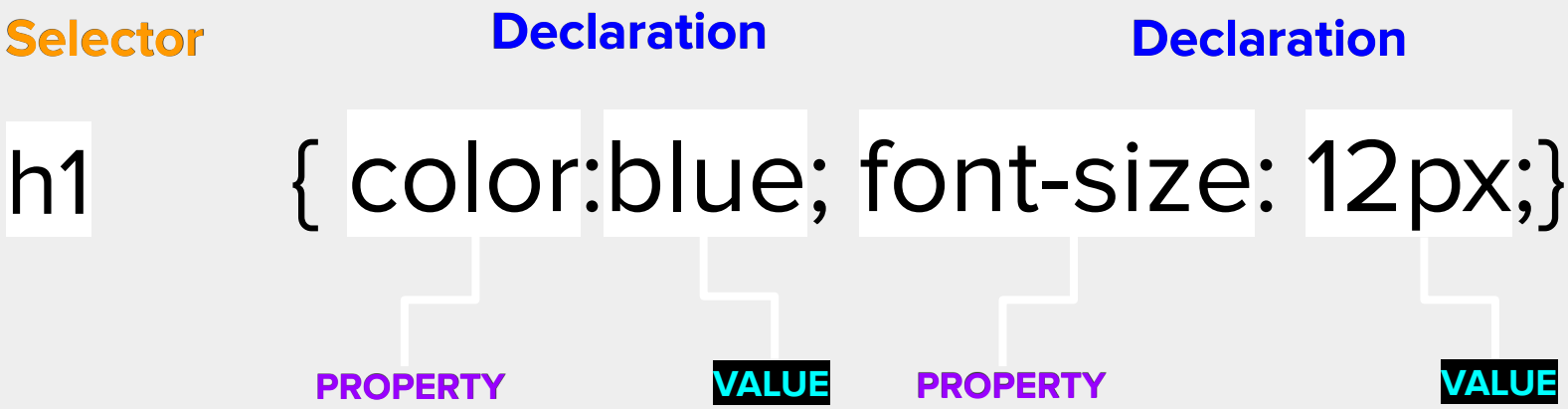
Different Methods of Applying CSS

- **Inline style sheet** is defined within a tag. Applies only to that particular occurrence of that tag.
- **Internal style sheet** (also called Embedded) is defined within the head section of a page. Applies to that page only.
- **External style sheet** defined in a separate file, hence external. Applies to all pages that link to the external style sheet.



CSS Syntax

A CSS rule has two main parts: a **selector**, and one or more **properties** set to a **value**:



- The selector is normally the HTML element you want to style.
- Each declaration consists of a property and a value.
- The property is the style attribute you want to change. Each property has a value.
- **IMPORTANT**: Always end your declaration with a semicolon

Commonly Used CSS Properties

Popular CSS Property/Values Usage Examples...

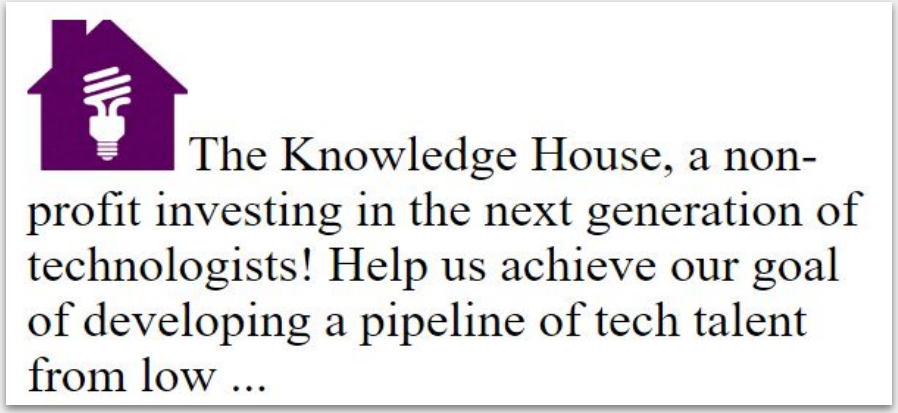
Property*	Description	Values	Example - Selector {Property:Value }
background-image	Sets an image as the background	url(?)	h2 {background-image: url(flowers.jpg) }
border	Sets the properties for the four borders	border-width border-style border-color	p {border-width: 3px 5px 3px 5px } h3 {border-style: double } h2 {border-color: red }
font-style font-size font-family	Sets the properties for a font	bold, italic, etc. px, em, pt, % arial, times, etc.	h4 {font-style: italic } div {font-size: 14px } body {font-family: verdana }
height	Sets the height of an element	px, em, pt, %	img {height: 200px } div {height: 90% }
list-style-type	Sets the style of the list item (li) bullet	disc circle square	ul {list-style-type: disc } ul {list-style-type: circle } ul {list-style-type: square }
margin	Sets margin properties	px, em, pt, %	h2 {margin: 15px } body {margin: 10% } div {margin: 1em }
padding	Sets padding properties	px, em, pt, %	h1 {padding: 10px } p {padding: 10%}

** Not all values for the properties listed are shown in this chart. As CSS evolves, new properties are emerging — refer to W3C.org.*

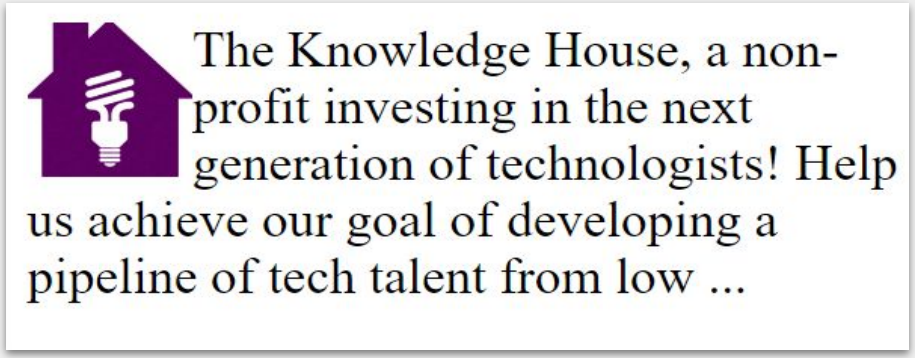
For an extended list of Properties visit http://www.w3schools.com/cssref/

Manipulating Images

Which of these look better?



1



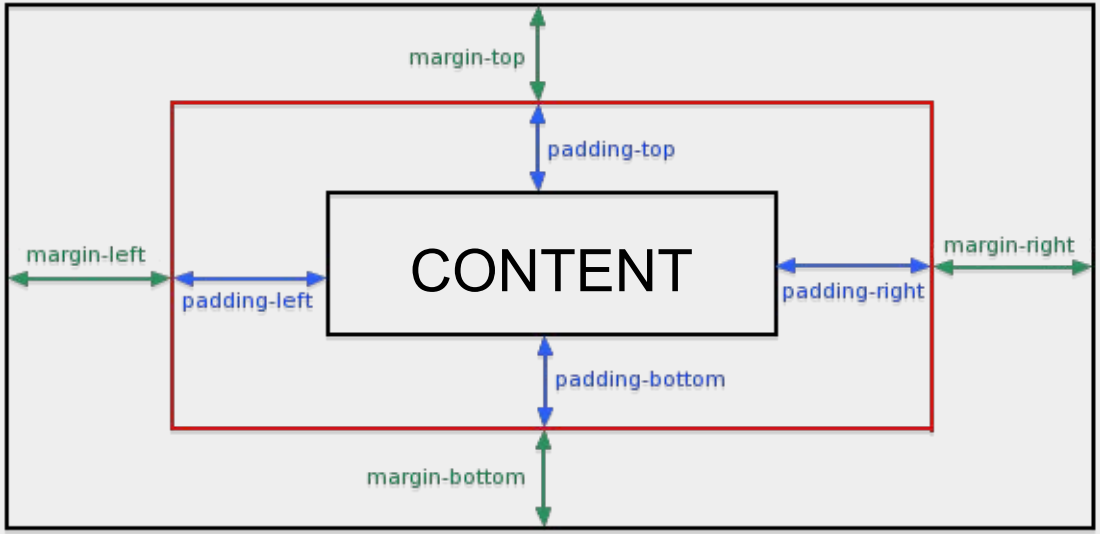
2

Padding vs Margin vs Border

Padding - Clears an area around the content. The padding is transparent

Border - A border that goes around the padding and content

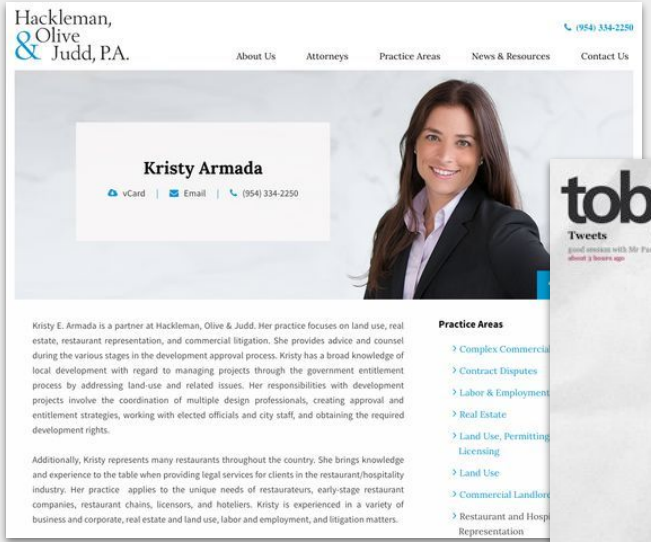
Margin - Clears an area outside the border.



Questions? Comments?
Emotional Outbursts?!

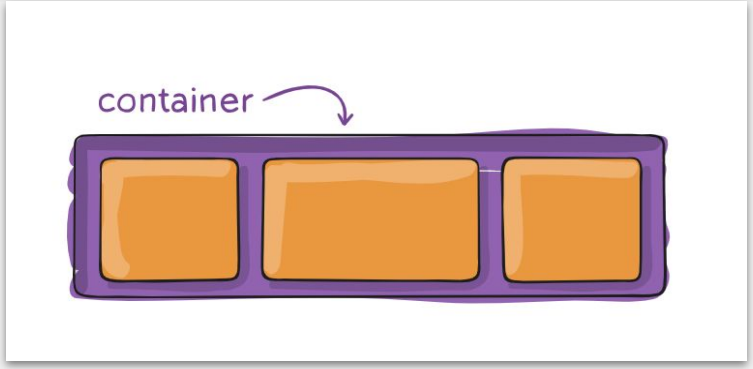
Website Layout

What do these Websites have in common?



CSS Flex

Since flexbox is a whole module and not a single property, it involves a lot of things including its whole set of properties. Some of them are meant to be set on the container (parent element, known as “flex container”) whereas the others are meant to be set on the children (said “flex items”).



CSS Flex

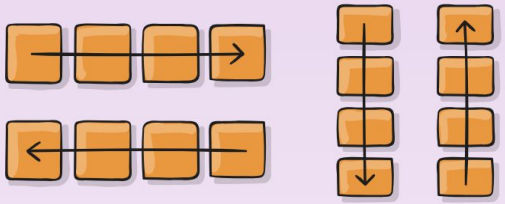
display

This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

```css
.container {
  display: flex; /* or inline-flex */
}
```

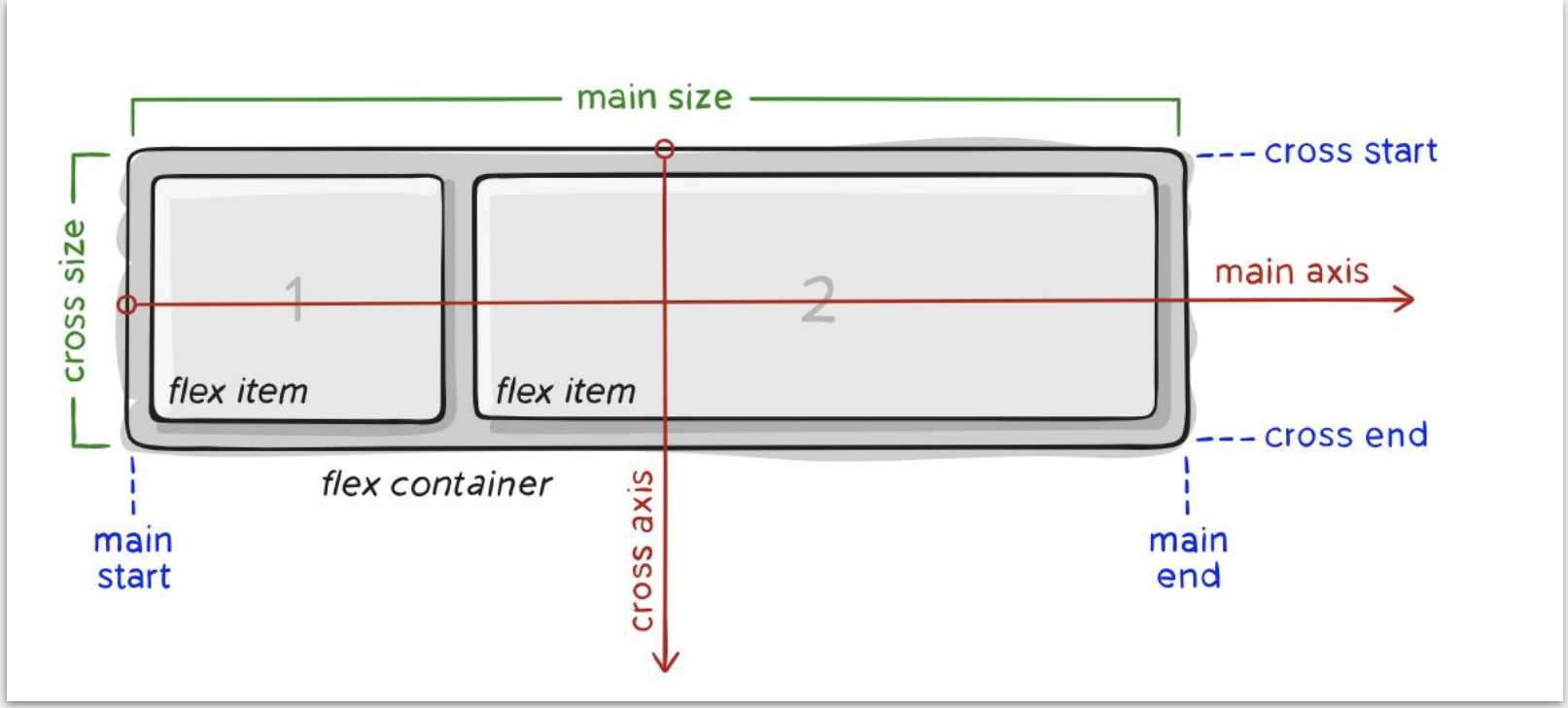
Note that CSS columns have no effect on a flex container.

flex-direction

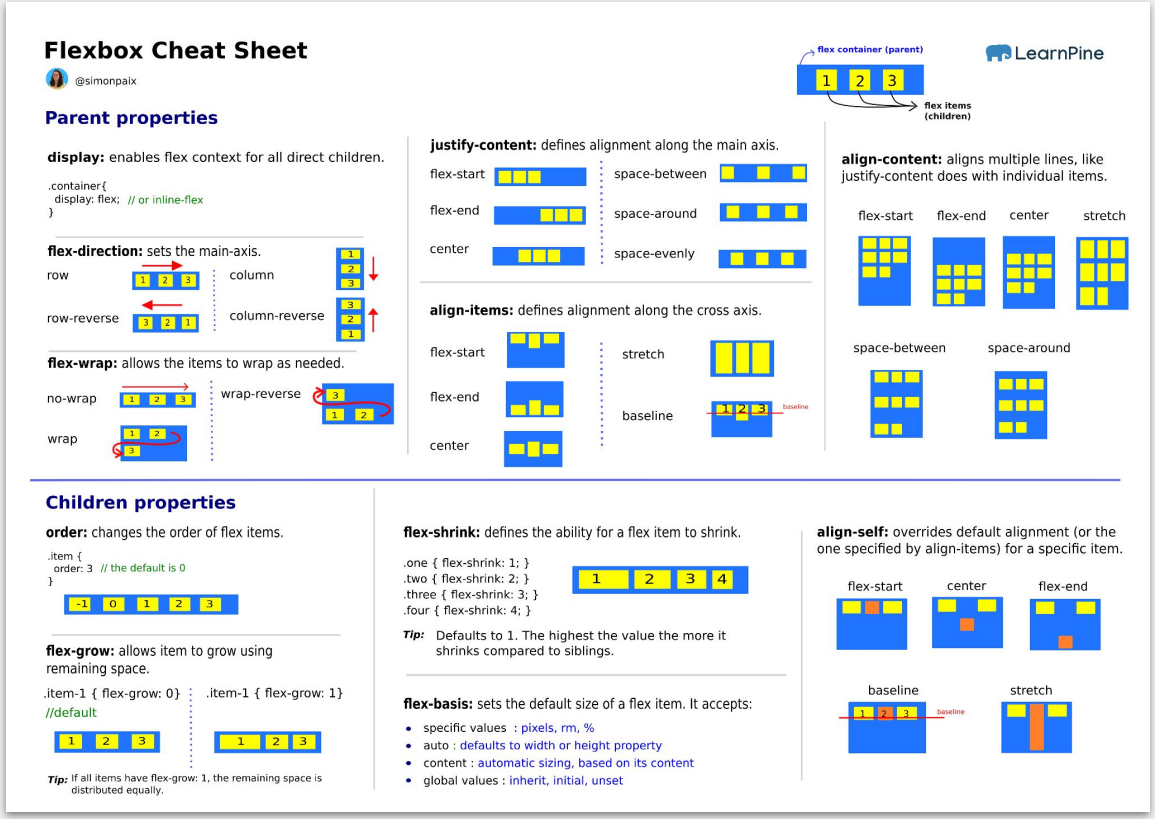


This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

CSS Flex


main size
cross start
cross size
main axis
1
2
flex item
flex item
cross end
flex container
main
start
cross axis
main
end

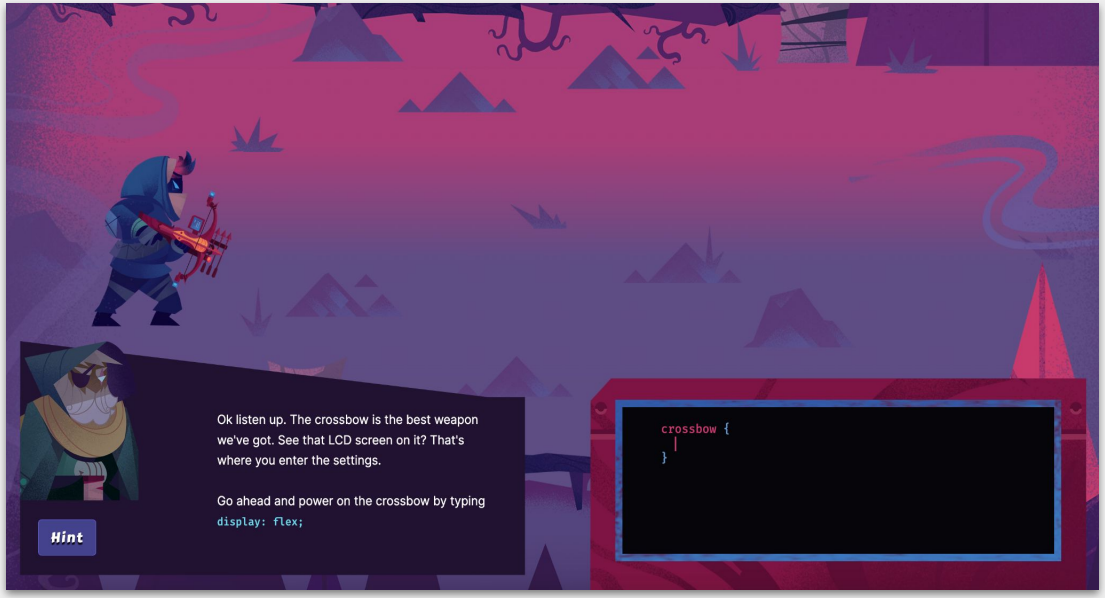
CSS Flex


Flexbox Cheat Sheet
@simonpaix
flex container (parent)
1 2 3
flex items (children)
LearnPine
Parent properties
display: enables flex context for all direct children.
.container{
display: flex; // or inline-flex
}
flex-direction: sets the main-axis.
row
column
row-reverse
column-reverse
flex-wrap: allows the items to wrap as needed.
no-wrap
wrap-reverse
wrap
justify-content: defines alignment along the main axis.
flex-start
flex-end
center
space-between
space-around
space-evenly
align-items: defines alignment along the cross axis.
flex-start
flex-end
center
stretch
baseline
align-content: aligns multiple lines, like justify-content does with individual items.
flex-start
flex-end
center
stretch
space-between
space-around
Children properties
order: changes the order of flex items.
.item {
order: 3 // the default is 0
}
-1 0 1 2 3
flex-grow: allows item to grow using remaining space.
.item-1 { flex-grow: 0}
//default
.item-1 { flex-grow: 1}
Tip: If all items have flex-grow: 1, the remaining space is distributed equally.
flex-shrink: defines the ability for a flex item to shrink.
.one { flex-shrink: 1; }
.two { flex-shrink: 2; }
.three { flex-shrink: 3; }
.four { flex-shrink: 4; }
1 2 3 4
Tip: Defaults to 1. The highest the value the more it shrinks compared to siblings.
flex-basis: sets the default size of a flex item. It accepts:
specific values : pixels, rm, %
auto : defaults to width or height property
content : automatic sizing, based on its content
global values : inherit, initial, unset
align-self: overrides default alignment (or the one specified by align-items) for a specific item.
flex-start
center
flex-end
baseline
stretch

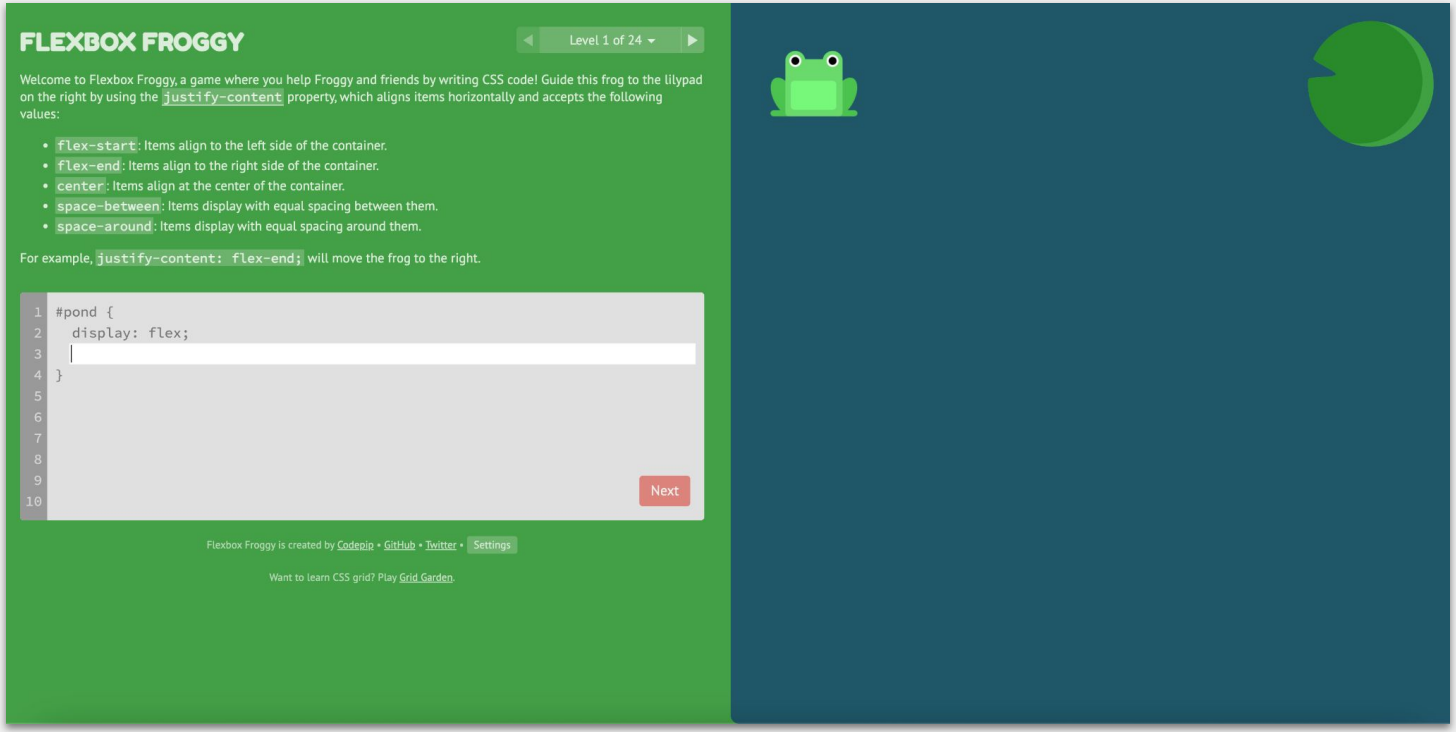
Exercise - Coin Flip

Let's get our hands on the keyboard and learn to Flex! Navigate to the "Flexbox Froggy" or "Flexbox Zombie"

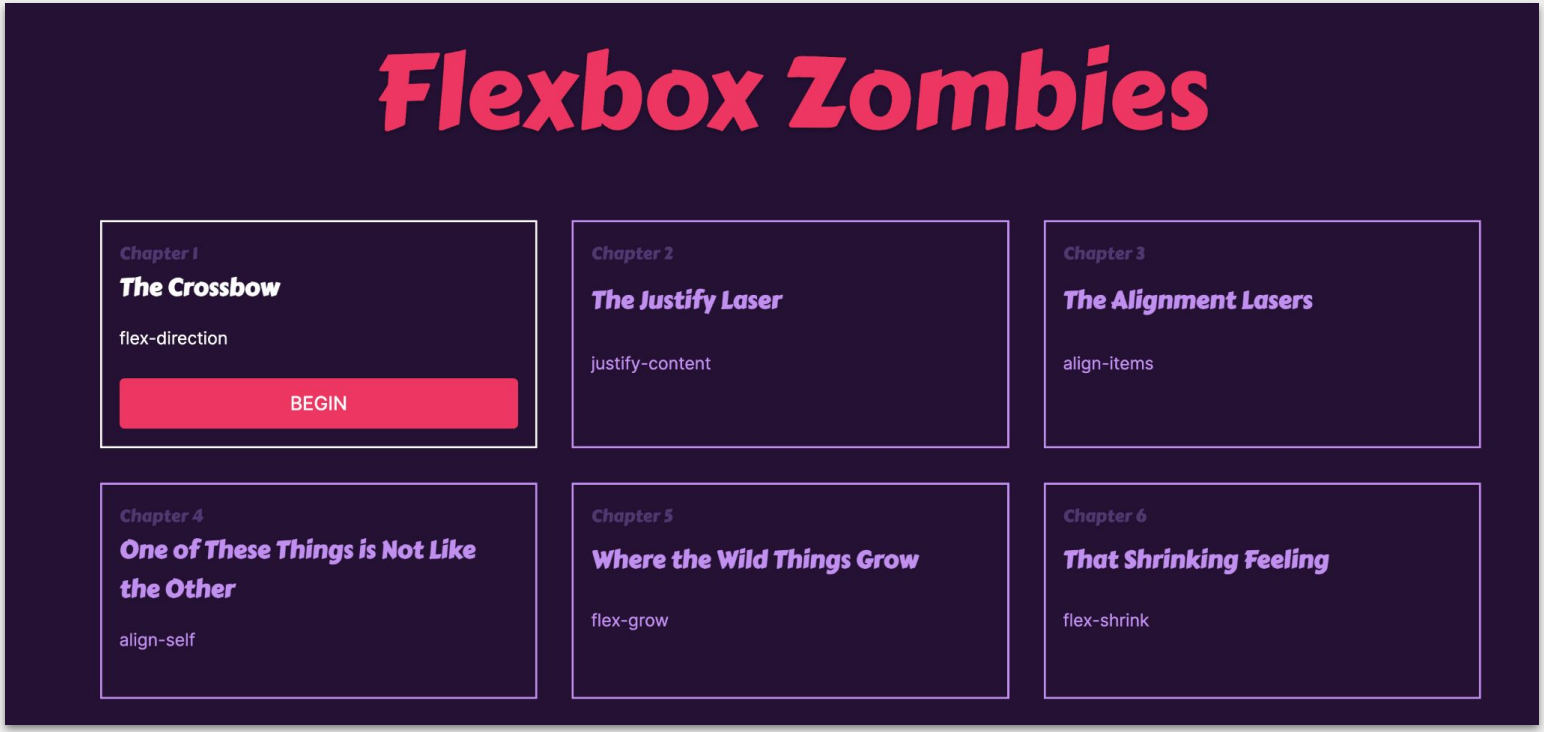
- Start with the first exercise
- Take 10 minutes to see how far you can get



Flexbox Froggy

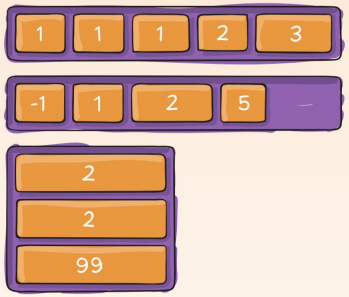


Flexbox Zombies



CSS Flex

order

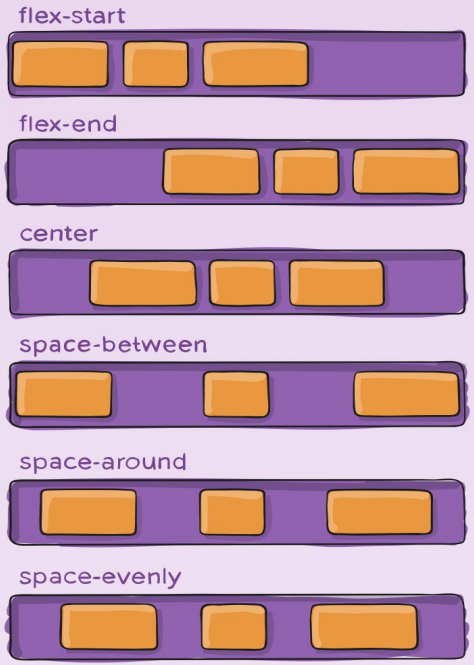


By default, flex items are laid out in the source order. However, the `order` property controls the order in which they appear in the flex container.

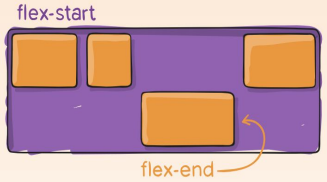
```css
.item {
  order: 5; /* default is 0 */
}
```

Items with the same `order` revert to source order.

justify-content



align-self



This allows the default alignment (or the one specified by `align-items`) to be overridden for individual flex items.

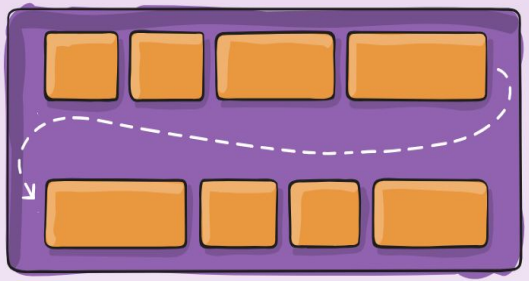
Please see the `align-items` explanation to understand the available values.

```css
.item {
  align-self: auto | flex-start | flex-end | center | ba
}
```

Note that `float`, `clear` and `vertical-align` have no effect on a flex item.

CSS Flex

flex-wrap



By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

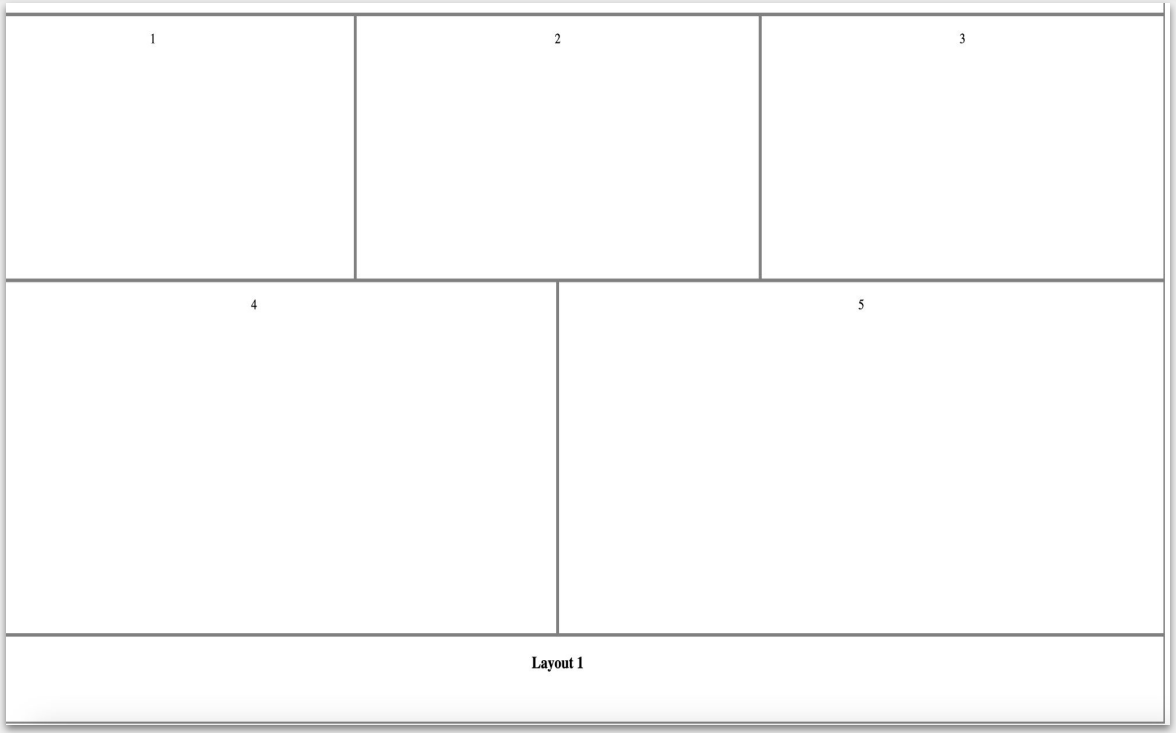
```css
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

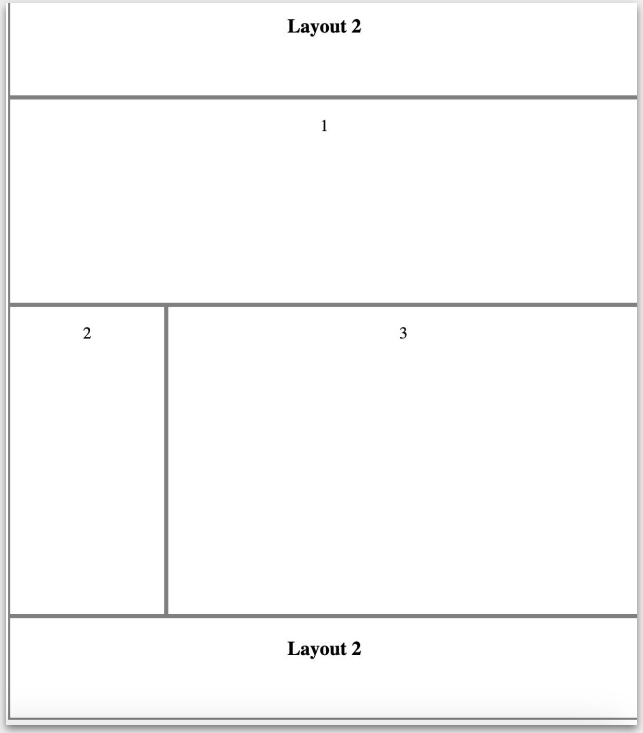
- `nowrap` (default): all flex items will be on one line
- `wrap`: flex items will wrap onto multiple lines, from top to bottom.
- `wrap-reverse`: flex items will wrap onto multiple lines from bottom to top.

Exercise

- **Objective**: Use your newfound knowledge of CSS Flex to execute at least one of the following 3 layouts
  - Each layout should be its own **Repl** with an `index.html` and `styles.css` page
  - Your CSS should be linked externally to the styles.css
  - Style it to make it easier to read
- **How to Submit Assignment:**
  - Place in a **Replit** project to submit for homework later
- **Share!**

CSS Layouts





CSS Layouts

Layout 3			
1	2	3	4

Debrief

1. What are your initial reactions to using CSS Flex?

2. How could you have used CSS Flex in past projects? What are some other ways to position elements on a page

Thank You

Thanks so much for your time and energy! Would you mind filling out this brief survey to tell us about your experience? Or consider leaving a Google Review!

Please leave any questions and comments here for us to respond to via email

- What more would you like to learn in this lesson?

Next Steps

Revisions