PER SCHOLAS

CSS Flexible Box Layout

PER SCHOLAS

Learning Objectives

This lesson provides an overview of the CSS Flexible Box Layout, commonly referred to as "flexbox." By the end of this lesson, learners will be able to:

- Apply the following flexbox properties and values to HTML pages:
  - `flex`
  - `justify-content`
  - `align-items`
  - `align-content`
  - `flex-wrap`
  - `flex-direction`
  - `flex-grow`
  - `order`

PER SCHOLAS

Table of Contents

PER SCHOLAS

CSS Flexible Box Layout

The CSS Flexible Box Layout (flexbox) was designed to achieve a complex layout model that the traditional **`float`** and **`position`** properties could not achieve easily. It is a one-dimensional layout method for arranging items into rows or columns. Items flex by expanding or shrinking to fit their available space.

A flexbox consists of various properties that help us define two types of boxes:

- Flex Containers have one or more flex items within them, and define how these flex items are positioned, ordered, and wrapped.
- Flex Items contain the content itself.

For example, a **`<div>`** container can be made into a flex container by providing the **`display`** property with a value of **`flex`** when styling the container. Any container inside is then considered a flex item.

4

Before the Flexbox Layout module, there were four layout modes:

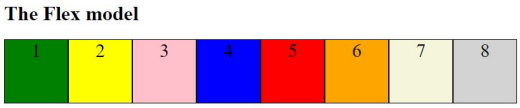
- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data
- Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

PER SCHOLAS

Activity: Basic Flexbox

We will use Flexbox to design the following box layout in HTML with CSS.



1) Create a file called **flexbox-examples.html**.
2) Create a ** ** with class container.
3) Create eight ** ** elements with class boxes.
   a) Each ** ** will have an **id** of **boxN**, where **N** is an incrementing number from 1 to 8, e.g. **box1**, **box2**, **box3**, etc.
   b) Each ** ** should contain its aforementioned incrementing number inside of it.
4) The CSS to the right can be added to a **<style>** element within the **<head>** of your HTML file.

```css
.container {
   display: flex;
}
.boxes {
   border: .1px solid;
   width: 99px;
   height: 99px;
   text-align: center;
   font-size: 30px;
}
#box1 { background-color: green; }
#box2 { background-color: yellow; }
#box3 { background-color: pink; }
#box4 { background-color: blue; }
#box5 { background-color: red; }
#box6 { background-color: orange; }
#box7 { background-color:beige; }
#box8 { background-color: lightgrey; }
```

Completed File Download

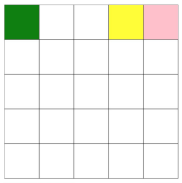
PER SCHOLAS

Flexbox: Auto Margins

You can use **auto margins** for main-axis alignment. Auto margins will take up all of the space that they can in their axis.

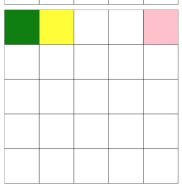
`margin-left: auto`

The example to the right shows the left margin set to auto on the yellow box, pushing it and the pink box to the far right of the container.



`margin-right: auto`

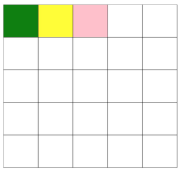
The example to the right shows the right margin set to auto on the yellow box, pushing the pink box to the far right of the container.



PER SCHOLAS

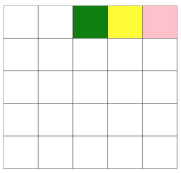
Flexbox Properties: Justify-Content

The CSS `justify-content` property defines how the browser distributes space between and around content items along the main axis of a flex container.



`flex-start`

The items are packed flush to each other toward the edge of the alignment container depending on the flex container's main-start side. This only applies to flex layout items. For items that are not children of a flex container, this value is treated like **`start`**.



`flex-end`

The items are packed flush to each other toward the edge of the alignment container depending on the flex container's main-end side. This only applies to flex layout items. For items that are not children of a flex container, this value is treated like **`end`**.

PER SCHOLAS

Flexbox Properties: Justify-Content (continued)

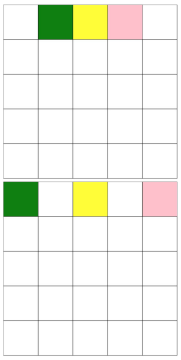
The CSS `justify-content` property defines how the browser distributes space between and around content items along the main axis of a flex container.

`center`

The items are packed flush to each other toward the center of the alignment container along the main axis.

`space-between`

The items are evenly distributed within the alignment container along the main axis. The spacing between each pair of adjacent items is the same. The first item is flush with the main-start edge, and the last item is flush with the main-end edge.



PER SCHOLAS

Flexbox Properties: Justify-Content (continued)

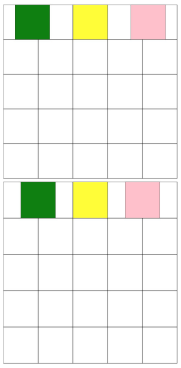
The CSS `justify-content` property defines how the browser distributes space between and around content items along the main axis of a flex container.

`space-around`

The items are evenly distributed within the alignment container along the main axis. The spacing between each pair of adjacent items is the same. The empty space before the first and after the last item equals half of the space between each pair of adjacent items.

`space-evenly`

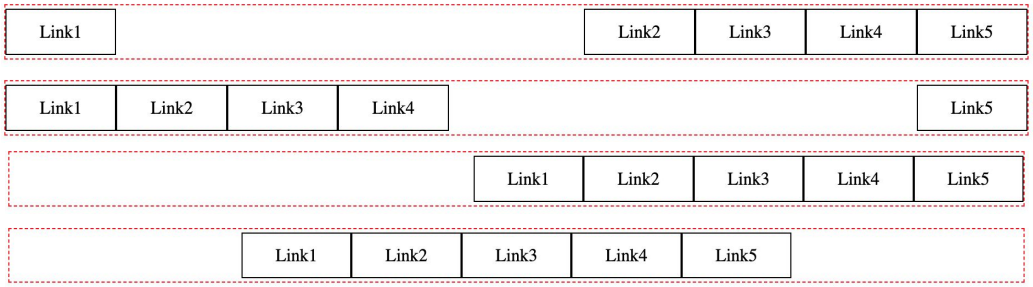
The items are evenly distributed within the alignment container along the main axis. The spacing between each pair of adjacent items, the main-start edge and the first item, and the main-end edge and the last item, are all exactly the same.



PER SCHOLAS

Activity: Navigation Bar Alignment – Part One

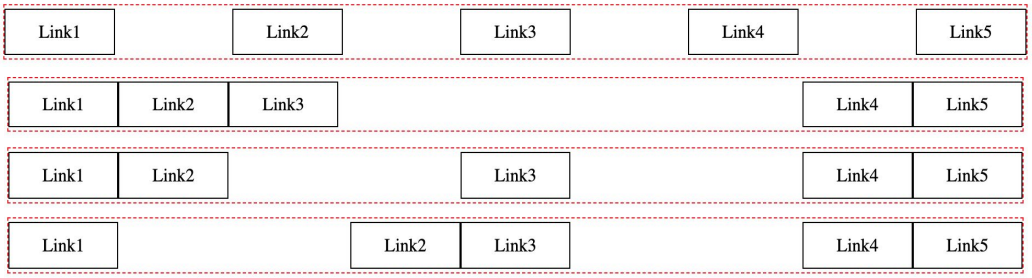
Use the provided navigation.html file to recreate the following layouts. Note that by adding **`display:flex`** to the **`<ul>`** element, the **`<ul>`** element becomes our flex container, and each **`<li>`** element is a flex item.



PER SCHOLAS

Activity: Navigation Bar Alignment – Part Two

Use the provided navigation.html file to recreate the following layouts. Note that by adding **`display:flex`** to the **`<ul>`** element, the **`<ul>`** element becomes our flex container, and each **`<li>`** element is a flex item.



PER SCHOLAS

Flexbox Properties: Align-Items

The CSS `align-items` property sets the `align-self` value on all direct children of the element. In flexbox, it controls the alignment of items on the cross axis.

`flex-end`

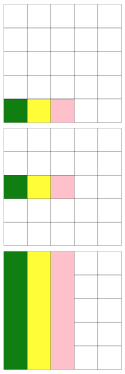
The cross-end margin edges of the flex items are flushed with the cross-end edge of the line.

`center`

The flex item margin boxes are centered within the line on the cross axis. If the cross-size of an item is larger than the flex container, it will overflow equally in both directions.

`stretch`

Flex items are stretched such that the cross-size of the item's margin box is the same as the line, while respecting width and height constraints.



PER SCHOLAS

Flexbox Properties: Align-Content

The CSS `align-content` property sets distribution of space between and around content items along a flexbox cross axis or a grid's block axis.

`space-between`

The spacing between each pair of adjacent elements is the same. The first item is flush with the start edge of the alignment container in the cross axis, and the last item is flush with the end edge of the alignment container in the cross axis.

`space-around`

The spacing between each pair of adjacent elements is the same. The empty space before and after the last item equals half of the space between each pair of items.

`space-evenly`

The spacing between each pair of adjacent items, the start edge and the first item, and the end edge and the last item, are all exactly the same.

PER SCHOLAS

Flexbox Properties: Flex-Grow

The CSS `flex-grow` property sets the flex grow factor of a flex item's main size. Elements within the container split the remaining space of their container by a ratio of their assigned **`flex-grow`** values.

`flex-grow: 1`

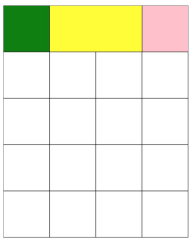
When all elements are assigned an equal value of one, they split the space of their container evenly.

`flex-grow: 2`

When one element is set to **`flex-grow: 2`**, in the case of the example to the right, it occupies 2/(1 + 2 + 1), or 50%, of the available space.

`flex-grow: ...`

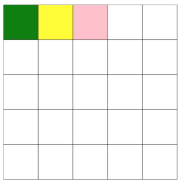
These values can be whatever you want to create the layout of your choice.



PER SCHOLAS

Flexbox Properties: Flex-Direction

The CSS `flex-direction` property sets how items are placed in the flex container, defining the main axis and the direction (normal or reversed).

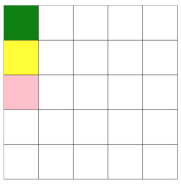


`row`

The flex container's main axis is defined to be the same as the text direction. The main-startpoint and the main-endpoint are the same as the content direction.

`row-reverse`

Behaves the same as **`row`**, but the main-startpoint and the main-endpoint are opposite.



`column`

The flex container's main axis is the same as the block axis. The main-startpoint and main-endpoint are the same as the before and after points of the writing mode.

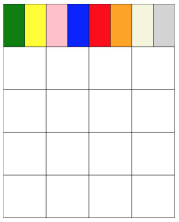
`column-reverse`

Behaves the same as **`column`**, but the main-startpoint and main-endpoint are opposite.

PER SCHOLAS

Flexbox Properties: Flex-Wrap

The CSS `flex-wrap` property sets whether flex items are forced onto one line or can wrap onto multiple lines. If wrapping is allowed, it sets the direction that lines are stacked.

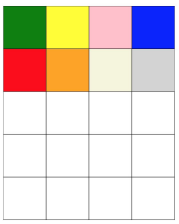


`nowrap`

Without flex wrapping, flex items can be *squished* into one line within their parent.

`wrap`

With wrapping enabled, if a box does not fit within the remaining space, it will be placed on a new line.



`wrap-reverse`

A reverse wrap behaves the same way as a regular wrap, but the cross-start and cross-end are permuted.

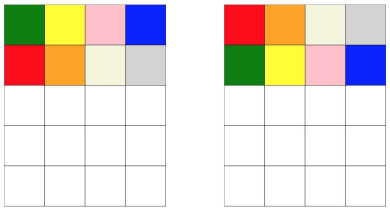
PER SCHOLAS

Flexbox Properties: Flex-Flow

The CSS `flex-flow` property specifies the direction of a flex container, as well as its wrapping behavior. **`flex-flow`** is a shorthand property that represents both **`flex-direction`** and **`flex-wrap`**.

A complete **`flex-flow`** value includes:

- A **`flex-wrap`** value:
  - **`nowrap`**
  - **`wrap`**
  - **`wrap-reverse`**
- A **`flex-direction`** value:
  - **`row`**
  - **`row-reverse`**
  - **`column`**
  - **`column-reverse`**

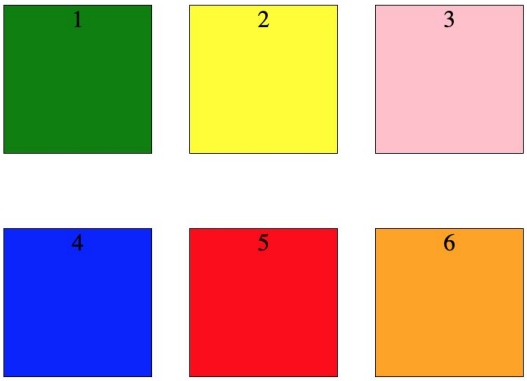


Initial Order Reversed

PER SCHOLAS

Activity: Flexbox Layout

Use what you have learned to create the following layout.



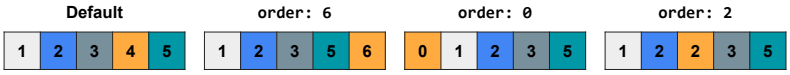
PER SCHOLAS

Flexbox Properties: Order

The CSS `order` property sets the order to lay out an item in a flex or grid container. Items in a container are sorted by ascending **`order`** values, and then by their order within the source code.

By setting an **`order`** value on elements within a flex container, you can organize those elements via CSS without editing the HTML structure.

For example, the following layouts can all be achieved with the exact same HTML source code by changing the **`order`** value for the orange block.



PER SCHOLAS

Flexbox Example: Basic Card

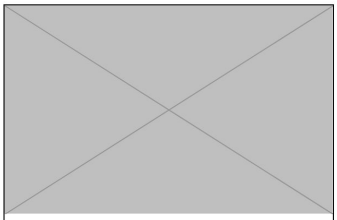
The following HTML and CSS show how to create a basic card structure using flexbox.

```html
<div class="card-body">
  <div class="img-container">
    <img src="/images/image-example.svg" alt="">
  </div>
  <h1>Heading</h1>
  <p>Lorem ipsum ..</p>
</div>
```

```css
.card-body {
    border: 1px solid black;
    width: 300px;
}

.img-container img {
    max-width: 100%;
    height: auto;
}

.card-body p, .card-body h1 {
    margin: 8px;
}
```



Heading

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aliquam amet distinctio ipsam sapiente! Aliquid corporis doloremque eius eveniet inventore libero magni, nulla quia quo ratione reiciendis saepe similique sint, sunt. lorem

PER SCHOLAS

Basic Card Code Break Down

- The HTML code contains two **`<div>`** elements. The first div is the container and has a class attribute of **`card-body`**. It holds the image and text for the card.
- The second div has a class attribute of **`img-container`** and holds the image for the card. Inside the **`img-container`** class is an image for demonstration purposes only.
- The CSS code starts with the **`card-body`** class, and has a **`border`** shorthand property and a **`width`** value of **`300px`**.
- The **`img-container`** class is using a descendant selector to select the **`img`** with class **`img-container`**. It has a **`max-width`** of one hundred percent (**`100%`**) with a **`height`** of **`auto`**, which will help with responsiveness.
- The **`card-body p`** and **`h`** tags both have margins of **`8px`** to give space to the text.

PER SCHOLAS

Activity: Flexbox Basic Cards

Using what you have learned, create the following layout using the card example from previous slides.

PER SCHOLAS

Practice Assignment: Design an Online Resume Using HTML and CSS

Please follow the link below to the practice assignment for designing an online resume using HTML and CSS.

- PA - 307.4.1 - Design an Online Resume using HTML and CSS
- You can also find this assignment on Canvas under the Assignments section.
- If you have technical questions while performing the activity, ask your instructors for assistance.
- When you are finished with your assignment, you can continue practicing your flexbox skills by visiting the following resource: https://flexboxfroggy.com/

Knowledge Check

- Name three of the five values for the **`justify-content`** property.
- What do auto margins do?
- Name two values for **`align-items`** property.
- Name two values for **`align-content`** property.
- What does the **`flex-grow`** property control?
- What does the **`flex-direction`** property do?
- What is **`flex-wrap`**, and how is it used?
- What is **`flex-flow`** a shorthand property for?
- Why is the **`order`** property useful for content organization?

PER SCHOLAS

Summary

In this lesson, we explored the CSS Flexible Box Layout, commonly referred to as “flexbox.” The CSS Flexible Box Layout is a powerful tool for organizing content on a webpage via CSS properties. Elements can be aligned along the main axis via the **`justify-content`** property, and aligned along the cross axis via the **`align-items`** and **`align-content`** properties, and include:

The flexbox properties and values can be applied to HTML pages:

- The **`flex-grow`** property can be used to control the relative size of items in their containers.
- The **`flex-direction`** and **`flex-wrap`** properties, combined within the shorthand **`flex-flow`** property, can be used to modify the way items order themselves within their containers.
- The **`order`** property can further modify the order of individual items within a container.

PER SCHOLAS

Questions?