Intro to Front End - Intro to FlexBox

Set up the Live Stream

Front End Fundamentals

- How are we finding CSS?

Intro to FlexBox

- Does anyone know what Flexbox is?
- Flexible Box! But Flexbox is more fun to say
- It is a CSS layout model that was created to replace a bunch of hacks

---

Objectives

- The relationship between elements (i.e. what is a parent, what is a child, what are siblings)
- How to declare a CSS block using tag names, classes and ids

Review this code - what are the relationships? How could we improve this? (semantics, indentation)

```
<div class="container">
  <div><img src="./angry.jpg" /></div>
  <div><img src="./beagle.jpeg" /></div>
  <div><img src="./derpy.jpeg" /></div>
  <div><img src="./grassy.jpeg" /></div>
  <div><img src="./regal.jpeg" /></div>
</div>
<div class="container">
  <div><img src="./angry.jpg" /></div>
  <div><img src="./beagle.jpeg" /></div>
  <div><img src="./derpy.jpeg" /></div>
  <div><img src="./grassy.jpeg" /></div>
  <div><img src="./regal.jpeg" /></div>
</div>
```

The Box Model NEW CODE PEN

- HTML layout - everything is boxes!

Whiteboard Box Model

Box Model comprises:

- Margin

- Padding
- Content
- Border
- Box model refers to how these properties relate to each other.
- We can change these using CSS but each element has a default property.

Live Code

- Put content in the box
- Change padding to 20px
- Change border to 5px solid black
- Change margin to 20px

The Box Model Contd

- The Box Model as a boring suburban neighbourhood

Height / Width This is your house (content)

Padding / Border The territory around the element that is limited by the border. Kind of like the yard and fence on your property. This is the padding and border.

Margin Empty space that separates this element from the surrounding elements. Similar to the trees that are still technically part of your property, but give you privacy from the neighbors. Meant to give you a buffer. This is the margin.

The Display Property NEW CODE PEN

- Every HTML element has a default display property which we can override using CSS.
- Block elements take up as much space as they can width wise and elements cannot be stacked side-by-side.
- Custom widths and heights can be applied
- Inline elements take up only as much space as they need
- We can't explicitly set a height and width on them.
- Inline-block has behaviour from both
- They are like inline elements but you can set a height and width on them.

Live Code

Add each of these properties to the h1, span, and img

```
h1 {
  border: 10px solid black;
  margin: 50px;
  padding: 50px;
  width: 100px;
```

```
  height: 100px;
}
```

FlexBox

- Flexbox is amazing! But it can be finicky at first
- It is a layout mode in CSS that means “flexible box model”.
- It overrides the default display settings of elements
- Allows developers to easily create grid layouts with spacing
- Without having to override the display properties of each individual element.
- Newer feature of CSS that solves some historical problems
- The tools CSS gave us to change the layouts of our page (floats!) haven’t always been that great, and often required a lot of work arounds.

Display: flex; NEW CODE PEN

- To start using flex box, we define the parent element to have a display property set to flex.

Live Code

Add this to the CodePen

```
article {
  display: flex;
}
```

Parent Properties NEW CODE PEN

- Flexbox has properties that can be applied to the parent and other properties that can be applied to the children.

Flex Direction

- Determines the direction in which the child elements will be arranged,
- Either in a row or column.
- The default is “row” which is horizontal but that can be modified to “column”

Live Code

Add this to the CodePen

```
.parent {
  display: flex;
}
```

```
flex-direction: row;
flex-direction: row-reverse;
flex-direction: column;
flex-direction: column-reverse;
```

Justify Content

- Determines how each child element will be laid out.
- Horizontal or vertical direction is determined by ‘flex-direction’
- But with this command we can set out the spacing between each element.
- Flex-End & flex-start: packs elements towards the end of the line or the start of the line
- Centre: places elements in the centre
- Space-Between: elements are evenly spaced with the first and last elements stuck to either end
- Space-Around: elements are arranged with equal space on both sides.
- Space-Evenly: elements are arranged so that the spacing between any two items (and the space to the edges) is equal

Live Code

Add this to the CodePen

```
justify-content: flex-end;
justify-content: flex-start;
justify-content: center;
justify-content: space-between;
justify-content: space-around; ** elements have equal space on either side
so the space between 2 elements will be double the space on either end
justify-content: space-evenly;
```

Align Items

- This property lays out items for the cross-axis.
- If the flex-direction is “row”, this is the “justify-content” property for the “column direction.

It accepts the following values:

- Flex-Start & Flex-End: element’s will be aligned with the start or end of the cross-axis
- Center: elements will be centred (either horizontally or vertically)
- Baseline: aligns along the baseline (ie: if there is text, the content will align along the bottom of the text

- Stretch: (default) elements will stretch to fill entire width or height but they will follow max-height and max-width css properties.

Live Code

Add this to the CodePen

```
align-items: flex-start;
align-items: flex-end;
align-items: center;
align-items: baseline;
align-items: stretch;
```

Flex Wrap NEW CODE PEN

- 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.

Live Code

Add this to the CodePen

```
flex-wrap: wrap;
flex-wrap: no-wrap;
flex-wrap: reverse;
```

Align Content

- If the parent contains multiple lines of child elements, and there is extra space leftover, we can use the “align-content” property to arrange the child elements.

- Flex-Start & Flex-End: arranges the children at the start or end of the parent container
- Center: arranges all the child elements in the center of container
- Space-Between: each line is distributed evenly with the first and last lines stuck to the top and bottom
- Space-Around: each line is equally distributed with equal space around each
- Stretch: (default) each line will stretch to take up the extra space

Live Code

Add this to the CodePen

```
align-content: flex-start;
align-content: flex-end;
align-content: center
align-content: space-between;
align-content: stretch;
```

Child Properties

- With Flexbox, you can also add properties to the children to control their behaviour in more detail.
- These properties do things like determine the size of individual children and give us much more control over the layout of our apps.

Order

- By default, child elements are arranged in their source order (ie: the order they appear in the HTML when reading top to bottom).

Live Code

Add this to the CodePen

```
order: 1;
order: 1;
order: -1;
order: 3;
order: -5;
order: 5;
order: 99;
```

Align-Self

- Allows the default arrangement, which was set by the parent container, on individual child elements.
- This property takes the same values as “align-items”:
- Flex-Start & Flex-End: element will be aligned with the start or end of the cross-axis
- Center: element will be centred (either horizontally or vertically)
- Baseline: aligns along the baseline (ie: if there is text, the content will align along the bottom of the text
- Stretch: (default) element will stretch to fill entire width or height but they will follow max-height and max-width css properties.

Live Code

Add this to the CodePen

```
align-self: flex-start;
align-self: flex-end;
align-self: center;
```

```
  align-self: baseline;
  align-self: stretch;
```

Flex Grow

- Tells a single child element if it grow if it needs to.
- The value it is given represents a proportion of the main axis that the child can take up.
- Setting all children to 1 will make them the same size and distribute the remaining space equally

Live Code

Add this to the CodePen

Remove all but 4 of the divs in the example and add the following values to the child elements

```
flex-grow: 2;
flex-grow: 3;
flex-grow: 1;
flex-grow: 0;
```

Change all the values to flex-grow: 1; to demonstrate that each element will be of equal size and evenly spaced.

Flex Shrink

- Allows you to set specific child elements that can get smaller if necessary.
- This is helpful if you need an item to be a very specific size.

Live Code

Add this to the CodePen

In the example, select one element to shrink

```
flex-shrink: 1;
```

Demonstrate that negative values are not valid

```
flex-shrink: -1;
```

Flex Shortcut

- The shorthand allows proportional values to set for flex-grow
- Flex-shrink can be added optionally.
- Flex basis determines the default size of an element before the left-over space is distributed.

- It can be set to a length or auto. In the shortcut it is optional and given as the third value.
- The default value is “ 0 1 auto”

Live Code

Add this to the CodePen

Add the following flex values the existing example

```
flex: 1;
flex: 2:
flex: 1
flex: 2;
```

Then show the students the full shortcut. Use the examples below in various arrangements

```
flex: 0 1 auto;
flex: 1 1 auto;
flex:  0 0 auto;
flex: 1 0 auto;
```

Practice

FLEXBOX FROGGY

Reminders

- Record Attendance
- Upload lecture
- Send out Notes
- Log hours on Clocktower