



Software Engineering

UNIT 1.2: WIREFRAMES, CSS-FLEX, CSS KEYFRAME, ANIMATION, CSS-TRANSFORM

Viewport Height

- When we apply a 100vh; it is like applying 100% to the element's height.



What is Flex-Box?

- Flex-box is a CSS3 layout mode intended to accommodate different screen sizes and different display devices
- Popular layouts can thus be achieved more simply and with cleaner code
- An efficient way to get the “float” look to your web pages.

Lets Watch a Tutorial as a class:

- <https://www.youtube.com/watch?v=MkzaON8NiK4>



CSS Layout: Intro to Flex

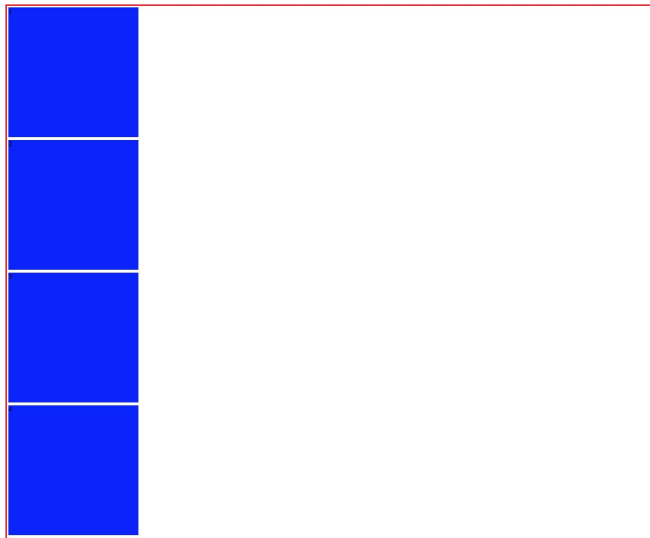
- Create 10 divs and assign them to a class
 - `<div class="child">1</div>`
- Wrap a container div around them
 - Name this container “parent”
- Begin styling the the divs and applying numbers (so we know what we’re looking at)

```
1 <div id="parent">
2   <div class="child">1</div>
3   <div class="child">2</div>
4   <div class="child">3</div>
5   <div class="child">4</div>
6   <div class="child">5</div>
7   <div class="child">6</div>
8   <div class="child">7</div>
9   <div class="child">8</div>
0   <div class="child">9</div>
1   <div class="child">10</div>
2 </div>
```

```
1 #parent{
2   border: solid red;
3 }
4 .child{
5   border:solid white;
6   width:300px;
7   height:300px;
8   background-color:blue;
9 }
0 }
```

CSS Layout: Intro to Flex

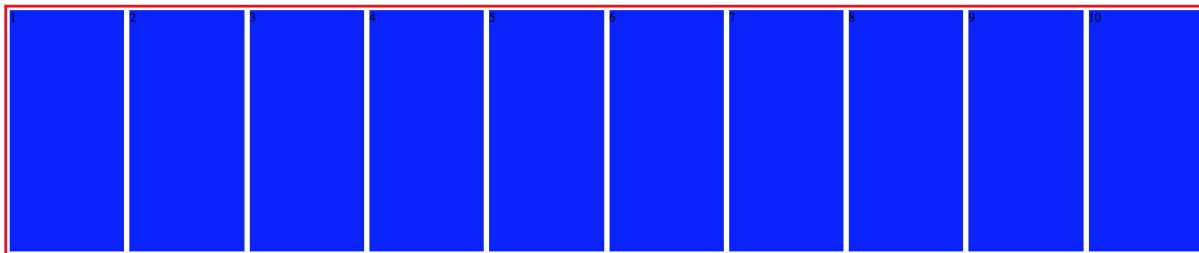
- By default, the divs will stack because HTML elements are block elements



CSS Layout: Intro to Flex

- Apply `display:flex;` to the parent.

```
1 #parent{  
2   border: solid red;  
3   display:flex;  
4 }
```



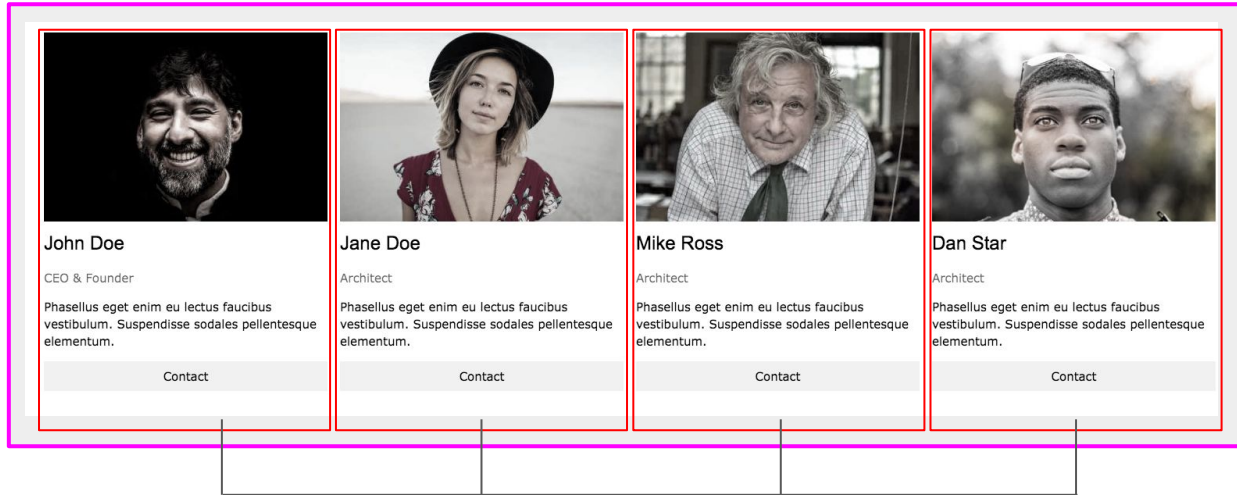
Note: all divs use up all the space available from Left to Right



CSS Layout: Sample Website

About

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.



This is
the
parent
div

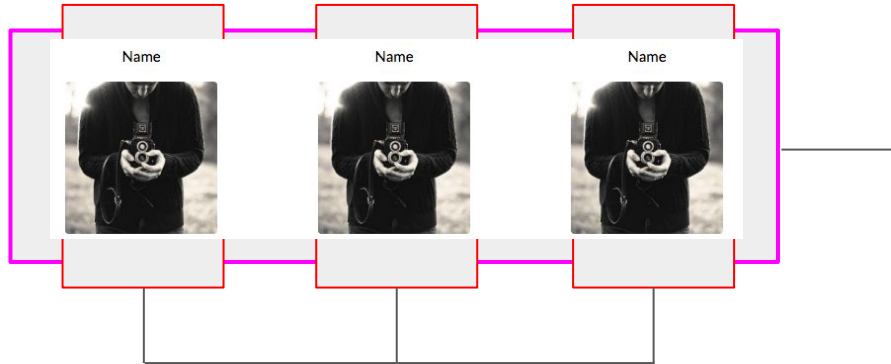
CSS Layout: Sample Website

This is the *parent div*

THE BAND

We love music

We have created a fictional band website. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.



These are all *child* divs with elements inside of them.

Exercise - code along

Open this code-pen: <https://codepen.io/mtavarez/pen/PBWVNa>

Lets go over it as a class!



Lab

Canvas: Week1_Day2_MorningLab



CSS: Inline-flex VS flex

- **Display: Inline-Flex** will wrap the around the size of the child elements in your container.
- **Display: Flex** will utilize all of the space from left to right.



CSS: Flex-Wrap


- The CSS **flex-wrap** property specifies whether flex items are forced into a single line or can be wrapped onto multiple lines.

Setting the container to wrap will cause the boxes to drop under one another once they don't fit


```
.container{
  display: flex;
  flex-wrap: wrap;
}
```




This is an example for flex-wrap:wrap



This is an example for flex-wrap:nowrap



This is an example for flex-wrap:wrap-reverse



CSS: Flex-Wrap

When you run out of space on one line; it breaks to the next line. Very similar to Float.

- give your “flex-items” a width:300px;

Notice how your divs work with the space it has on the single line despite that your screen size.



CSS: Flex-Wrap

Now apply flex-wrap on the parent element.

- The default is
flex-wrap: nowrap;
- give it a value of “wrap”

Notice how your divs now wrap around and break to the next line. if 300px cannot fit in the space left over, it simply goes to the next line.



CSS: Reverse

- Applying reverse to any of your css flex values will reverse the direction in which your browser read the content.
- For example:
 - `flex-wrap: wrap-reverse;`

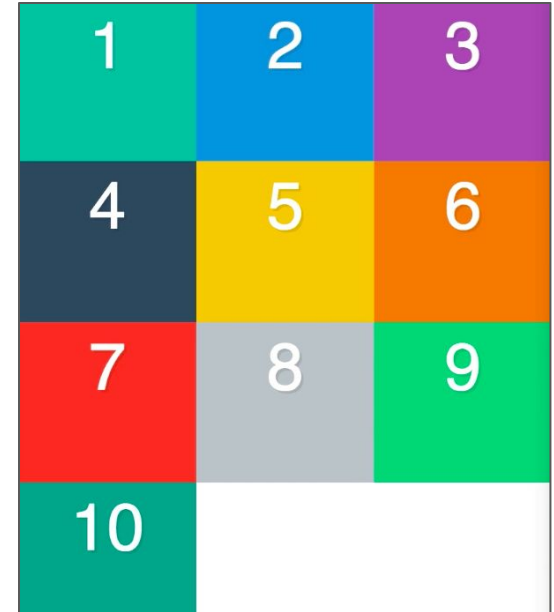
Notice how the divs now begin from 10.



CSS: Percentages vs Pixels

- When using pixels it will not evenly distribute it to all of your divs.
- Apply a width of 33.3333%

Notice how the space is now being utilized entirely from left to right.





CSS: Margin vs Padding vs Border

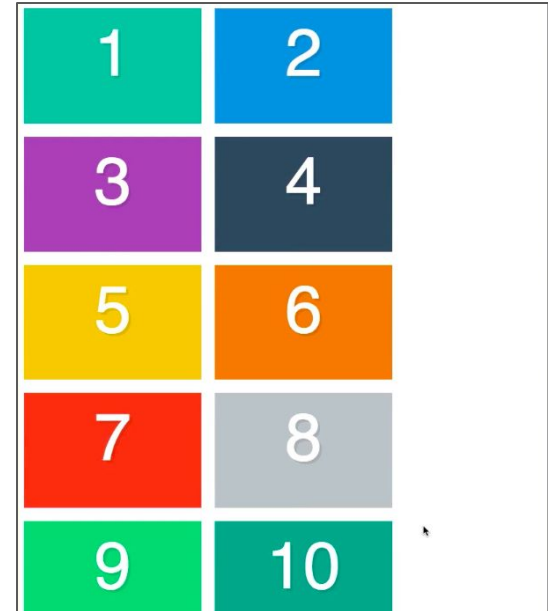
- Margin is not part of the flexible-box model. When using padding or border, it will not evenly distribute it to all of your divs.
- Apply a width of 33.3333% &
- Apply a margin of 10px;

You can fix this by apply the calc function and eliminating the px being added.

For example:

width: calc(33.333% - 20px);

margin:10px;



Flex-order

- create 5 divs and wrap them around a container
- assign them all to a class, and individually id them
- change the order by manipulating the order property.

```
32 .container {  
33   display: flex;  
34 }  
35  
36 .box {  
37   flex: 1;  
38 }  
39  
40 .box3 {  
41   order: 1;  
42 }  
43
```

Justify-content:flex-start;

- Allows you to align your items on the page using the flexible box model.

by default, justify-content is set to : flex-start





Justify-content:flex-end;

- This method maybe a look you want for a navigation bar aligned to the end.

`justify-content: flex-end`



Justify-content - center

- Easily center your divs using the flexible box model by applying :

`justify-content: center;`



Justify-content: space-between;

- Notice how the space gets evenly distributed between the divs.

justify-content: space-between;





CSS: Flex-Direction

- **flex-direction:row**
 - **(Main Axis)**
 - this is the default value for flex-direction. This will align the divs side by side
 - flex-direction:column
 - **(Cross Axis)**
 - This value will have your elements live as line items. This will stack your divs on top of each other.



Flex-direction

flex-direction: row;



Flex-direction

Try changing the
direction of your
parent element
and applying the
space between

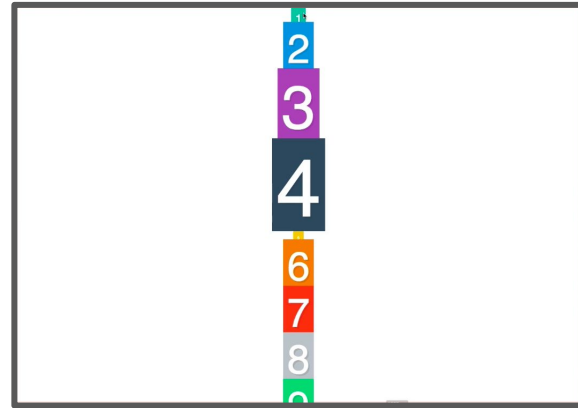


Main Axis vs Cross Axis

Justify-content aligns your items on the Main Axis (left to right)



align-content aligns your items on the cross Axis (Top to bottom)



What Are Media Queries

- The @media query is a CSS3 module that allows developers to manipulate the set of css rules assigned to their HTML elements based upon screen resolution (e.g. smartphone screen vs. computer screen).

```
nav{
  color: green;|
  text-decoration: none;
  padding: 3px;
  display: block;
}

@media (max-width: 699px) and (min-width: 520px) {
  nav{
    padding-left: 30px;
    background: url(pic.jpg) ;
  }
}
```

Breakpoints

Below is an image of the most popular desktop screen resolutions in 2016. This will help you identify when to use breakpoints.

| Screen width | Smallest screen height | In use worldwide Source: W3C, Jan 2016 |
|----------------|------------------------|---|
| 1024 px | 768 px | 3% |
| 1280 px | 800 px | 11% |
| 1360 px | 768 px | 2% |
| 1366 px | 768 px | 35% |
| 1440 px | 900 px | 6% |
| 1600 px | 900 px | 6% |
| 1680 px | 1050 px | 3% |
| 1920 px | 1080 px | 20% |
| 2560 px and up | 1440 px | 1% |

Breakpoints

Usable site space in browser, based on screen width:

| Device screen width | Web site safe area = screen resolution minus browser interface and vertical scrollbar. | |
|---------------------------|--|---------|
| | width | height |
| iphone portrait (320 px) | 310 px | 352 px |
| iphone landscape (480 px) | 468 px | 202 px |
| ipad portrait (768 px) | 760 px | 920 px |
| ipad landscape (1024 px) | 1010 px | 660 px |
| screen : 1024 px | 989 px | 548 px |
| screen : 1152 px | 1117 px | 644 px |
| screen : 1280 px | 1245 px | 580 px |
| screen : 1360 px | 1325 px | 548 px |
| screen : 1366 px | 1331 px | 548 px |
| screen : 1440 px | 1405 px | 680 px |
| screen : 1600 px | 1565 px | 680 px |
| screen: 1680 px | 1645 px | 825 px |
| screen : 1920 px | 1885 px | 855 px |
| screen : 2560 px | 2525 px | 1220 px |

Breakpoints

- Most commonly used breaks points are:

CSS Breakpoints: where & how much?

You don't need to write mediaqueries for every possible screen resolution. To keep things simple you could target three groups:

- smaller than 768 px
- smaller than 1024 px
- larger than 1024 px

Setting rules for screen resolutions between (0px-768px) and (width:1024px) and (width:1024px) can help you cover all of your basis.

Resource

