CSS

-Vishal Avalani

Units

- **rem** - rem units are relational to the font-size value of the HTML tag. For example, if the font size of the HTML tag is 16px (that is the default size for an html document), then 1rem unit is equal to 16px. That makes .5rem=8px, 2rem=32px, etc.
- **em** - em units are similar to rem units, but whereas a rem unit always references the HTML tag, an em unit is relational only to it's nearest defined parent element. For example, if the div wrapper for a callout is set to font-size:20px, then any child element set to 1em would be the equivalent of 20px, .5em=10px, 2em=40px, etc.

Must read: https://zellwk.com/blog/rem-vs-em/ | https://www.sitepoint.com/power-em-units-css/

```css
h1 { font-size: 20px } /* 1em = 20px */
p { font-size: 16px } /* 1em = 16px */

h1 {
  font-size: 2em; /* 1em = 16px */
  margin-bottom: 1em; /* 1em = 32px */
}

p {
  font-size: 1em; /* 1em = 16px */
  margin-bottom: 1em; /* 1em = 16px */
}

h1 {
  font-size: 2rem;
  margin-bottom: 1rem; /* 1rem = 16px */
}

p {
  font-size: 1rem;
  margin-bottom: 1rem; /* 1rem = 16px */
}
```

Contd..

- **px** - Pixels are defined as a single point in a graphic image, and are often tied to viewport resolution. If a viewport is 1600x900, that typically means that there are 1600px pixel columns & 900 pixel rows, with a pixel in each cell. This definition is maybe overly simple when factoring in dpi or pixel density, but can stand as reference for the most part.
- **vw/vh** - the vw (view-width) and vh (view-height) units are relational to the viewport size, where 100vw or vh is 100% of the viewport's width/height. For example, if a viewport is 1600px wide, and you specify something as being 2vw, that will be the equivalent of 2% of the viewport width, or 32px.

Interesting use of vw/vh - full screen background

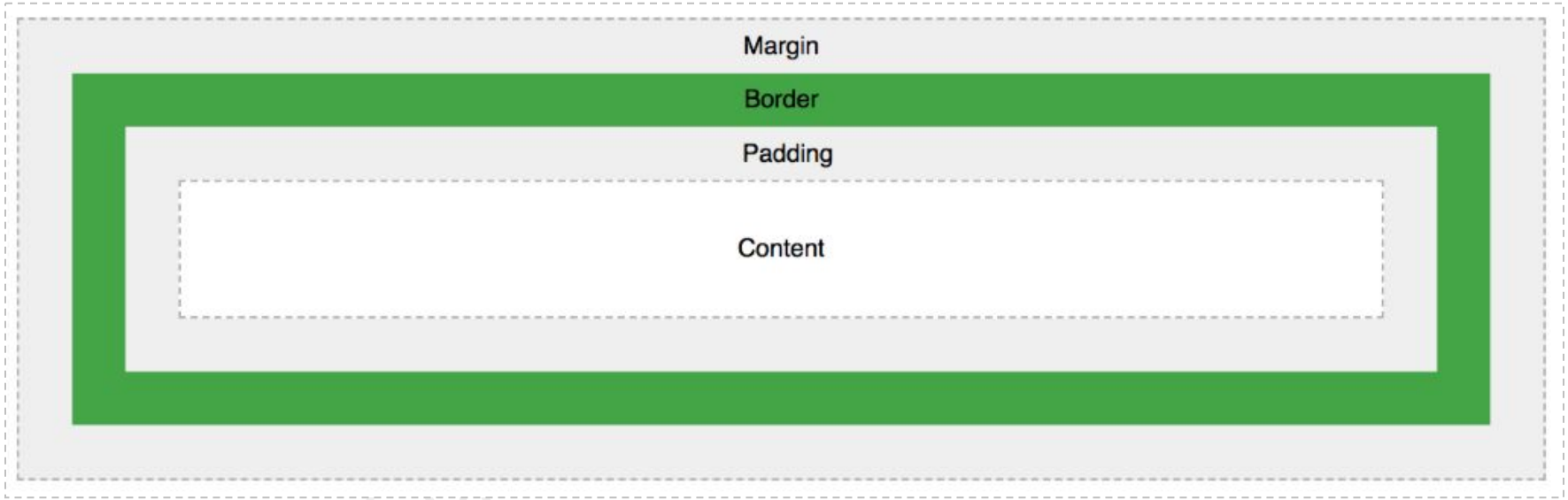
```
<div class="fullscreen a">
  <p>a<p>
</div>
```

You can achieve a full-width background image section using the CSS below:

```
.fullscreen {
  width: 100%;
  height: 100vh;
}
```

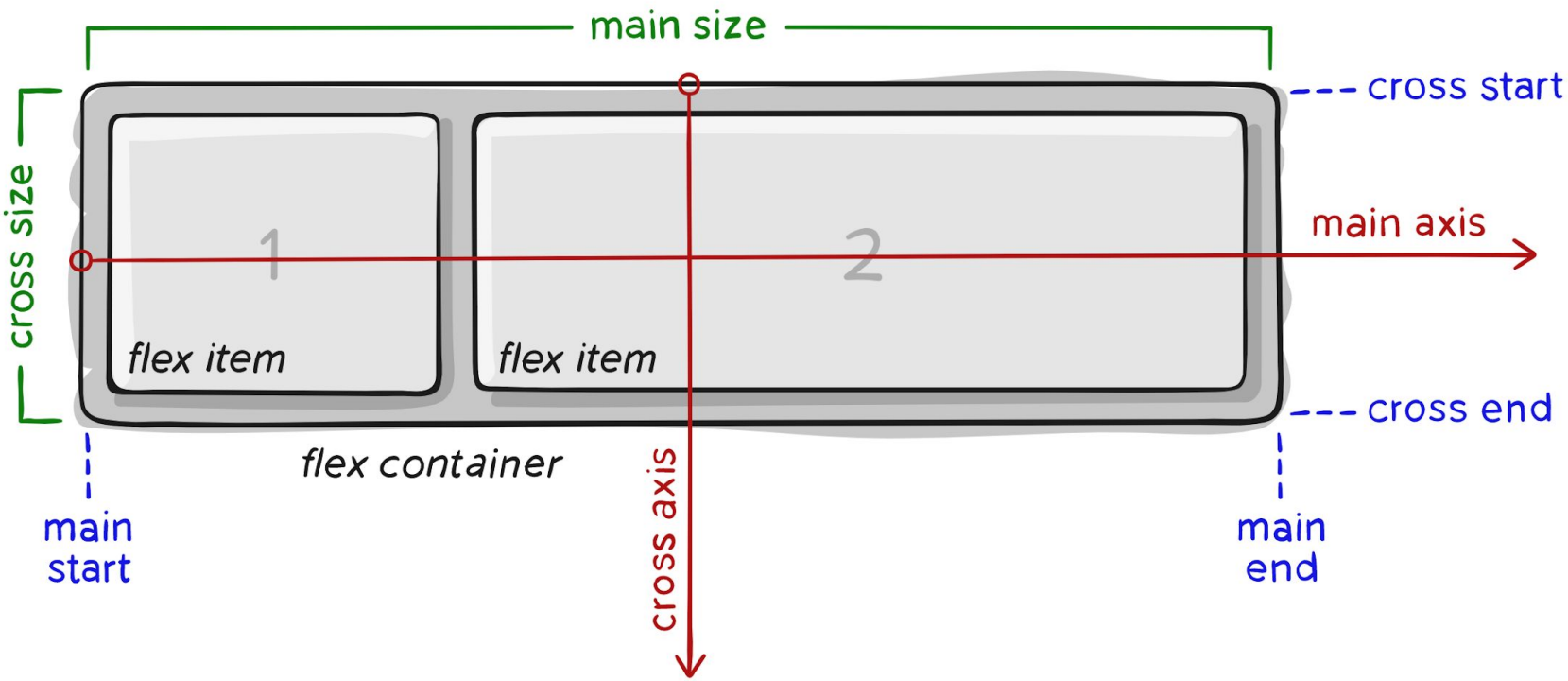
```
.a {
  background: url('path/to/image.jpg') center/cover;
}
```

CSS Box Model

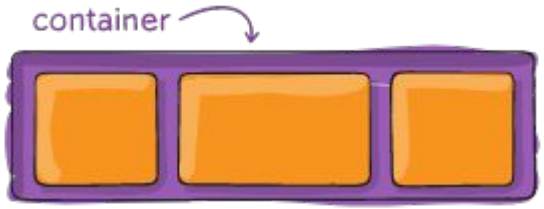


Excellent website: http://learnlayout.com/ | http://learnlayout.com/box-model.html

Flexbox

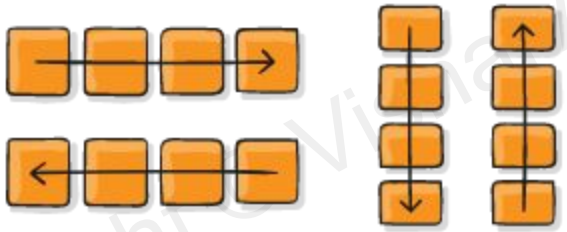

main size
cross start
cross size
main axis
1
2
flex item
flex item
cross end
flex container
cross axis
main
start
main
end

Parent Properties

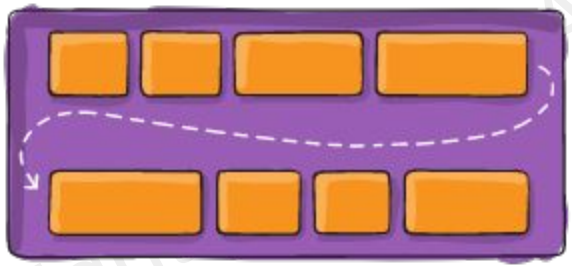


- display: flex;
- flex-direction: row | row-reverse | column | column-reverse;
- flex-wrap: nowrap | wrap | wrap-reverse;
- justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;
- align-items: stretch | flex-start | flex-end | center | baseline;
- align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch;
- flex-flow: <‘flex-direction’> || <‘flex-wrap’>

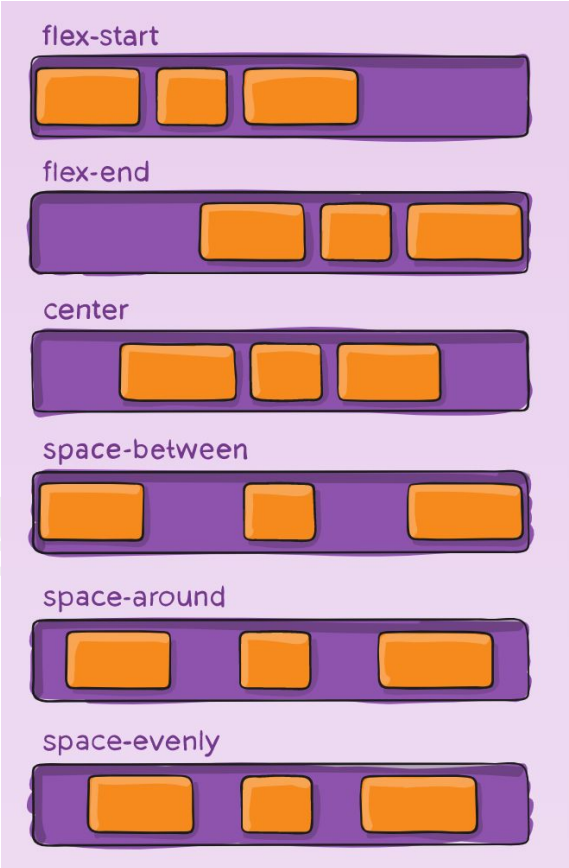
flex-direction



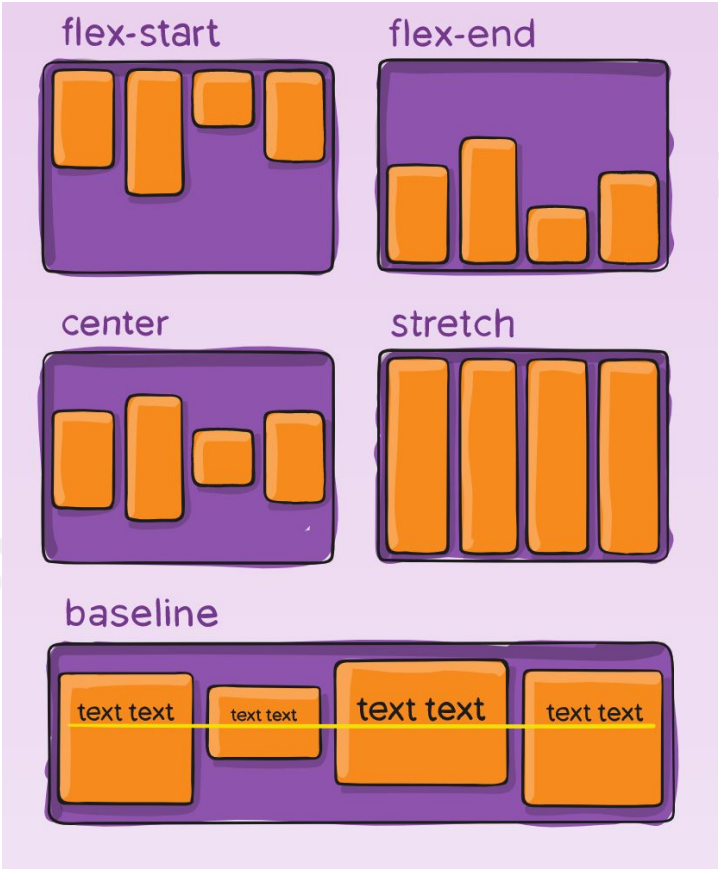
flex-wrap



justify-content



align-items

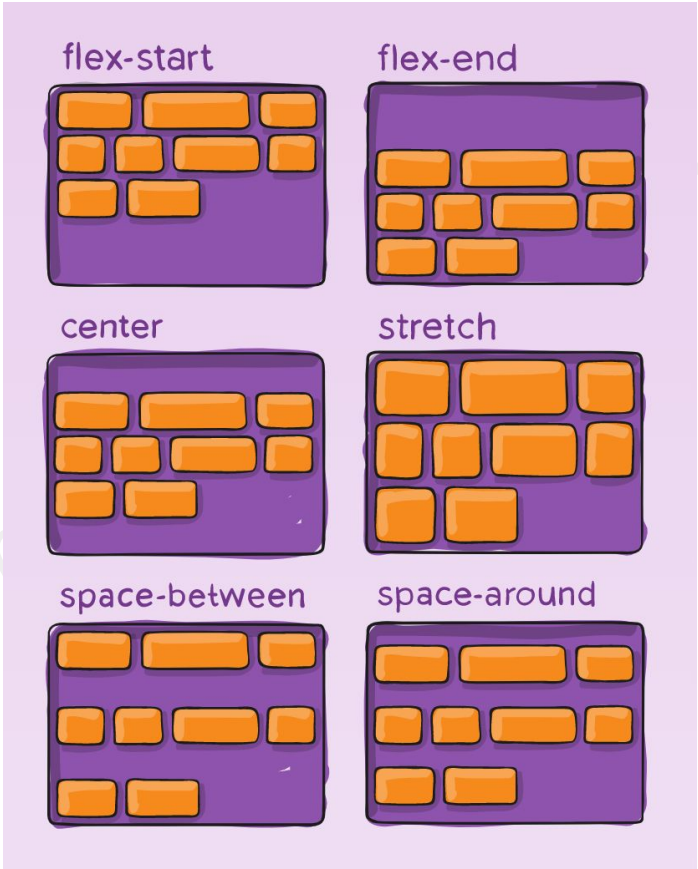

flex-start
flex-end
center
stretch
baseline
text text
text text
text text
text text

align-content

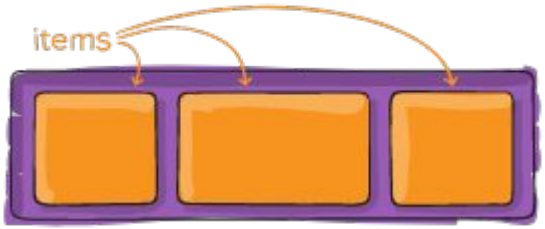
This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

Note: this property has no effect when there is only one line of flex items.

align-content

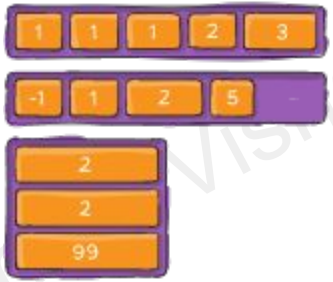


Children Properties



- order: <integer>;
- flex: none | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]
- align-self: auto | flex-start | flex-end | center | baseline | stretch;
- flex-grow: <number>;
- flex-shrink: <number>;
- flex-basis: <length> | auto;

order



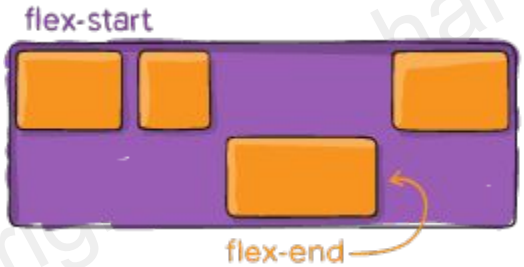
flex

This is the shorthand for flex-grow, flex-shrink and flex-basis combined. The second and third parameters (flex-shrink and flex-basis) are optional. The default is 0 1 auto, but if you set it with a single number value, it's like <number> 1 0.

It is recommended that you use this shorthand property rather than set the individual properties. The shorthand sets the other values intelligently.

align-self

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



flex-grow

flex-shrink

This defines the ability for a flex item to shrink if necessary.

flex-basis

This defines the default size of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword. The auto keyword means "look at my width or height property" (which was temporarily done by the main-size keyword until deprecated). The content keyword means "size it based on the item's content"

Only across main axis – px % or em. Main Axis is very important!

Flexbox Reference Links

- https://medium.com/@js_tut/the-complete-css-flex-box-tutorial-d17971950bdc
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- http://learnlayout.com/flexbox.html
- https://www.w3schools.com/css/css3_flexbox.asp
- Responsive Image gallery: https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_image_gallery
- Responsive Website: https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_website2