ADVANCED CSS

FLEXBOX

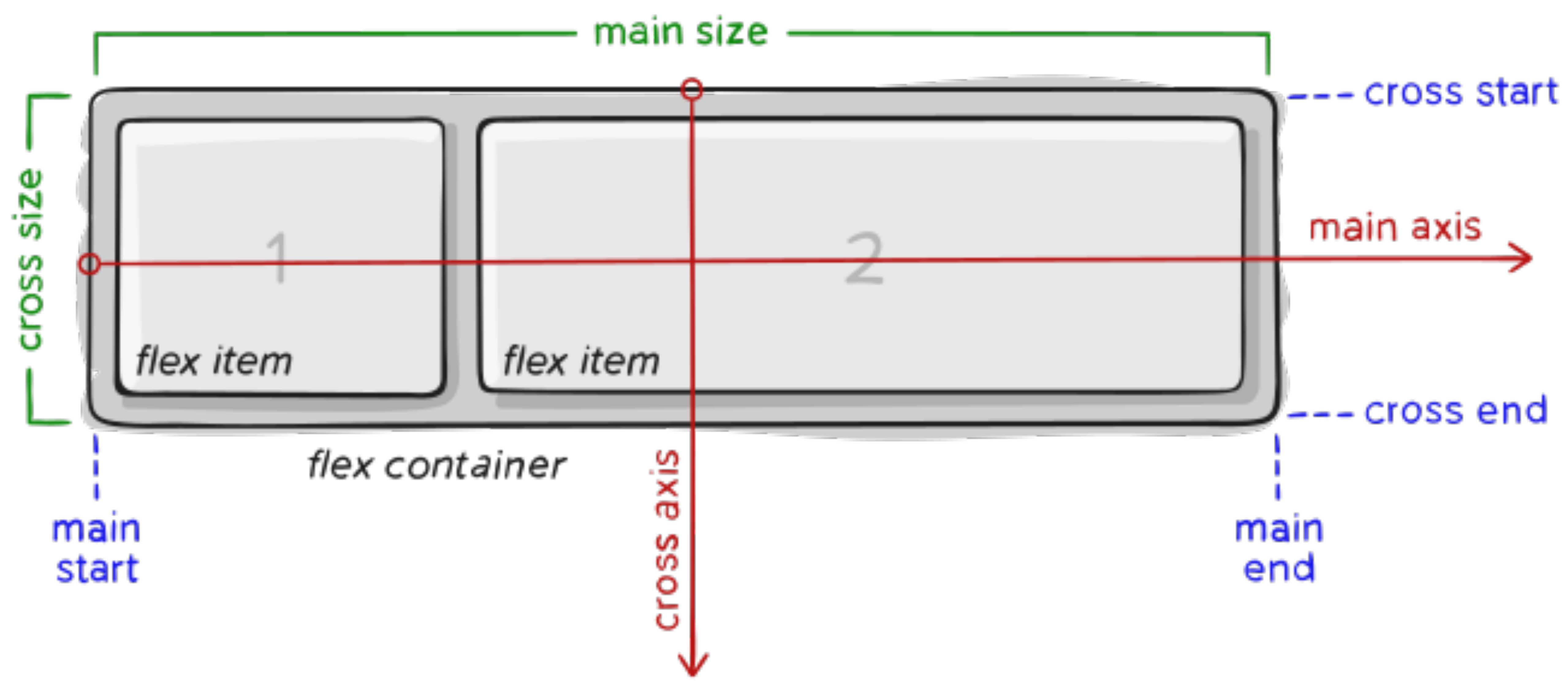
WHAT IS FLEXBOX?

Flexbox is a layout model which aims to make it easier to lay out and align elements dynamically.

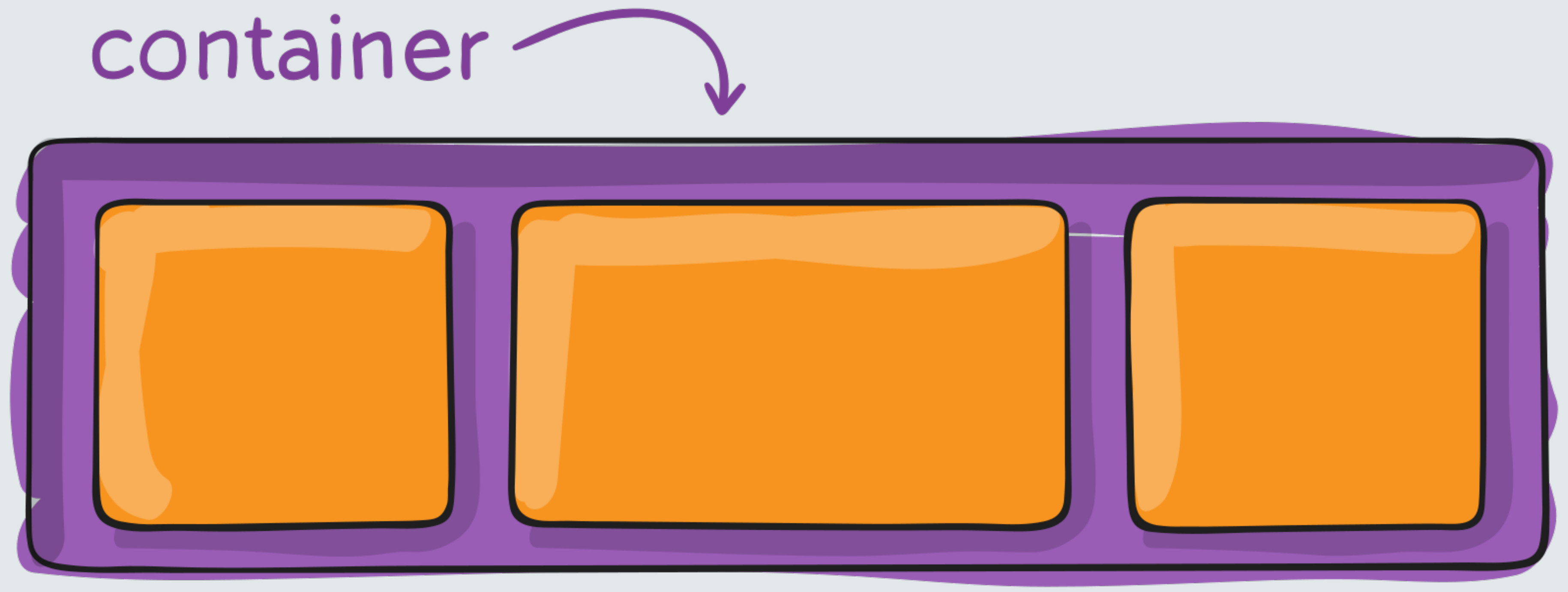
Main Idea: Containers have the ability to adjust their content dynamically.

Flexbox is **direction-agnostic**: can accommodate both **horizontal** and **vertical** layouts.

TERMINOLOGY



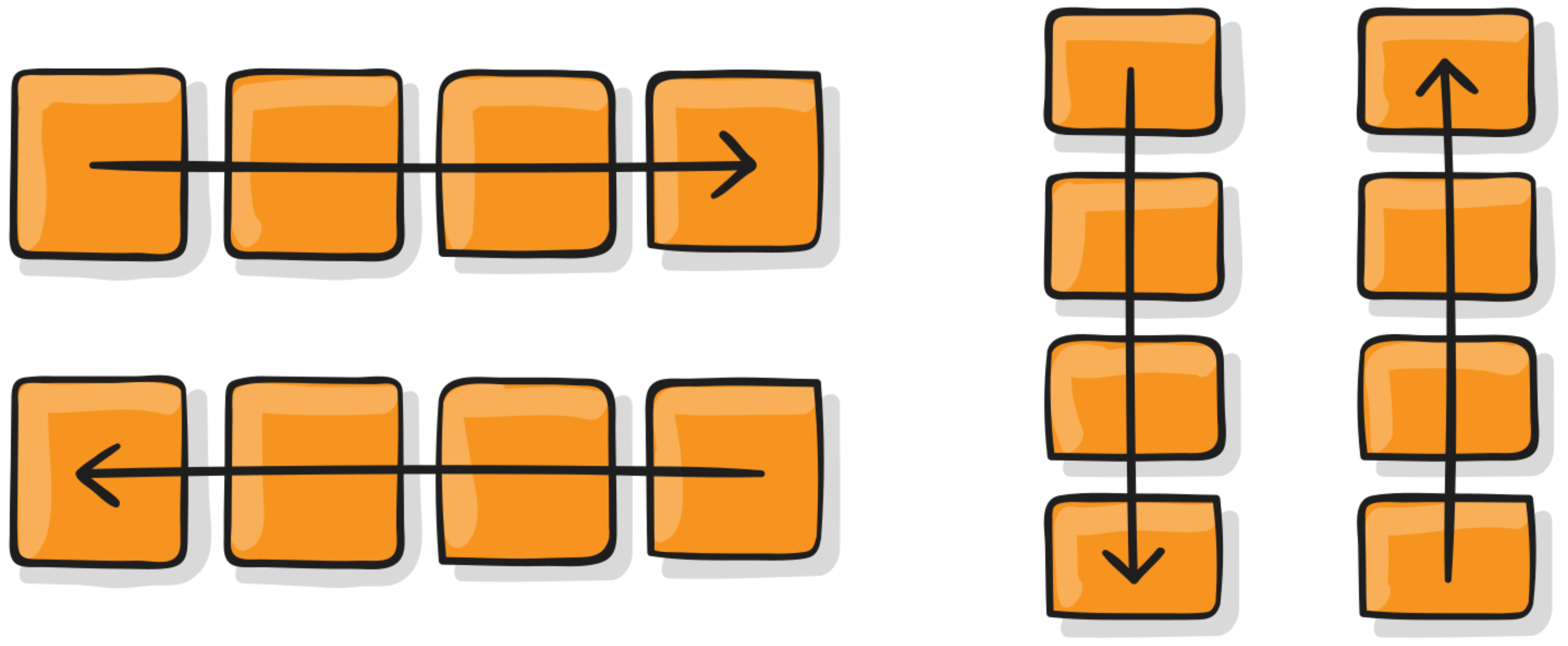
Container/Parent Properties



DISPLAY

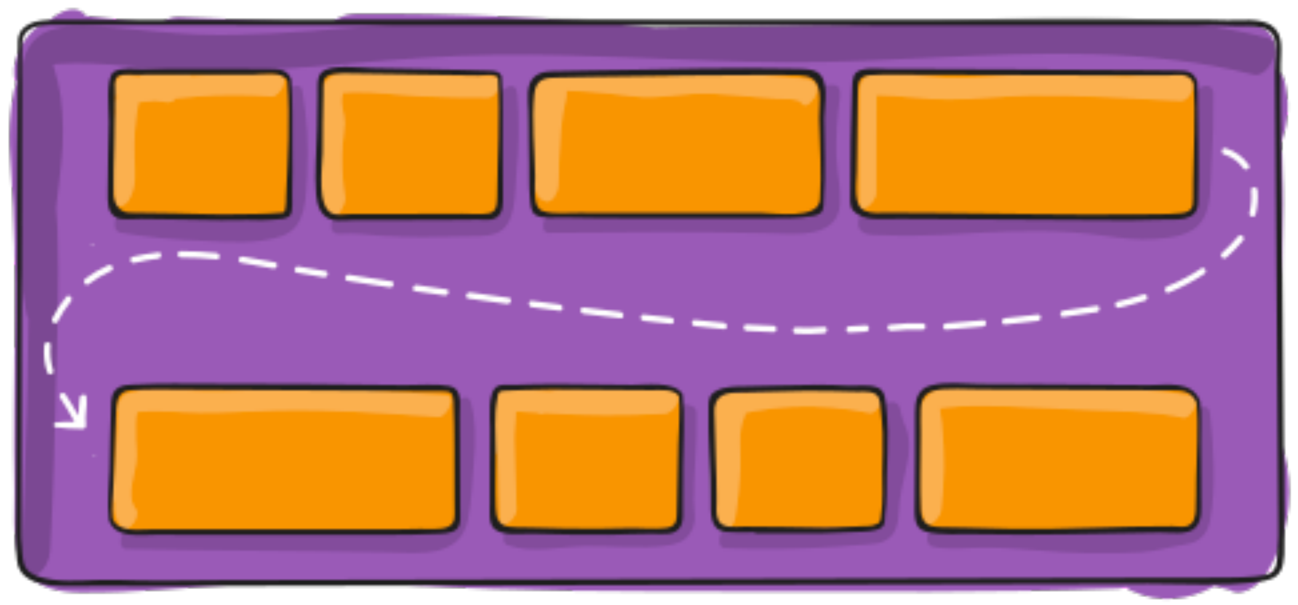
```css
.container {
  display: flex;
}
```

FLEX-DIRECTION



```
.container {
  flex-direction: row | row-reverse | column | column-reverse;
}
```

FLEX-WRAP

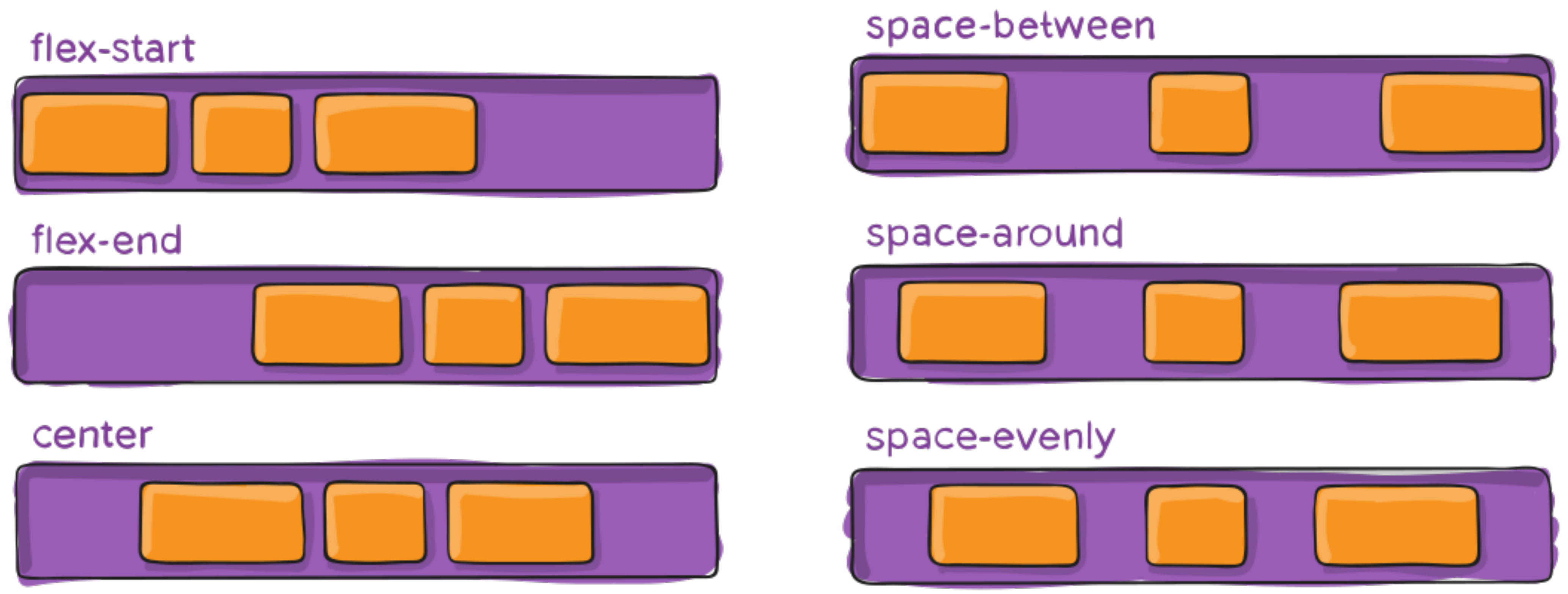


```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

FLEX-FLOW

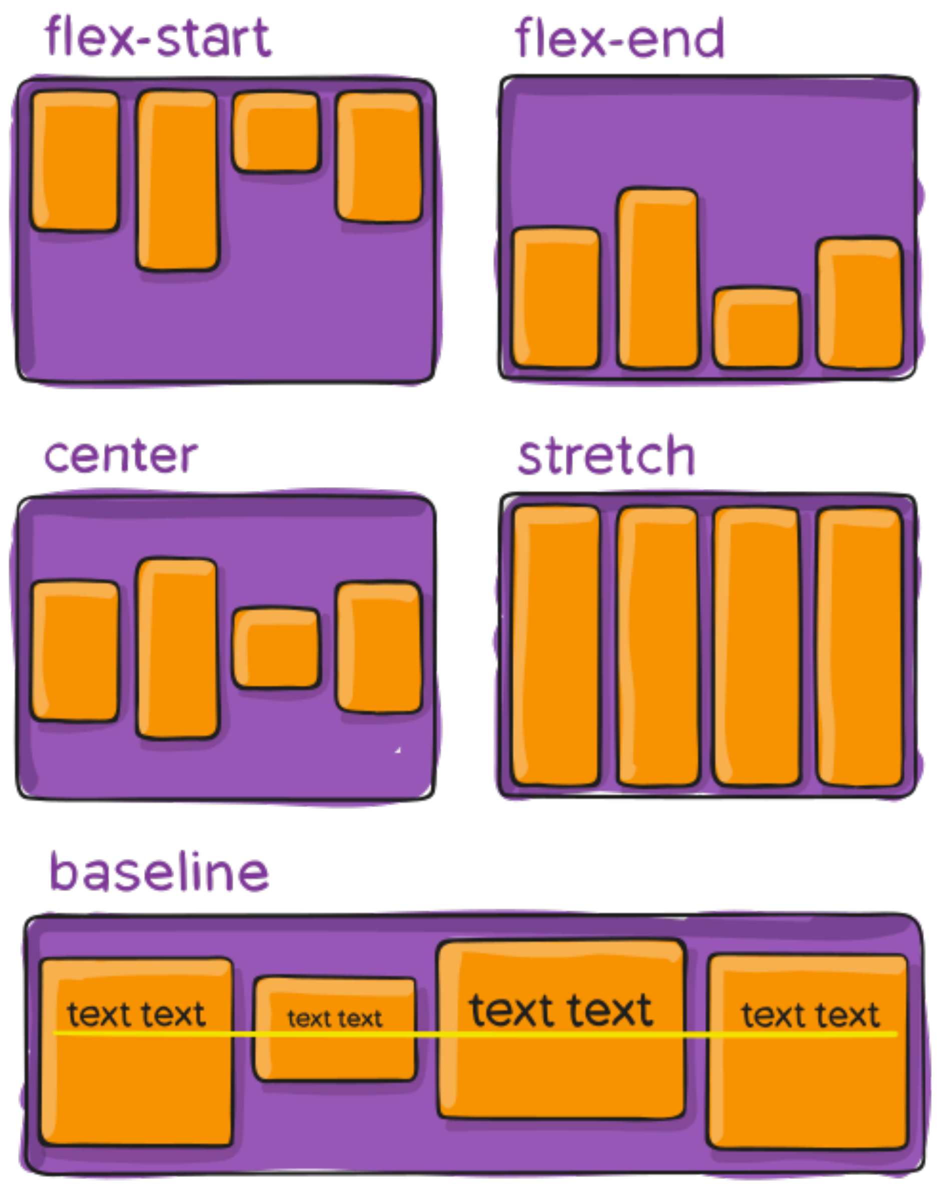
```
.container {
  flex-flow: <flex-direction> || <flex-wrap>;
}
```

JUSTIFY-CONTENT



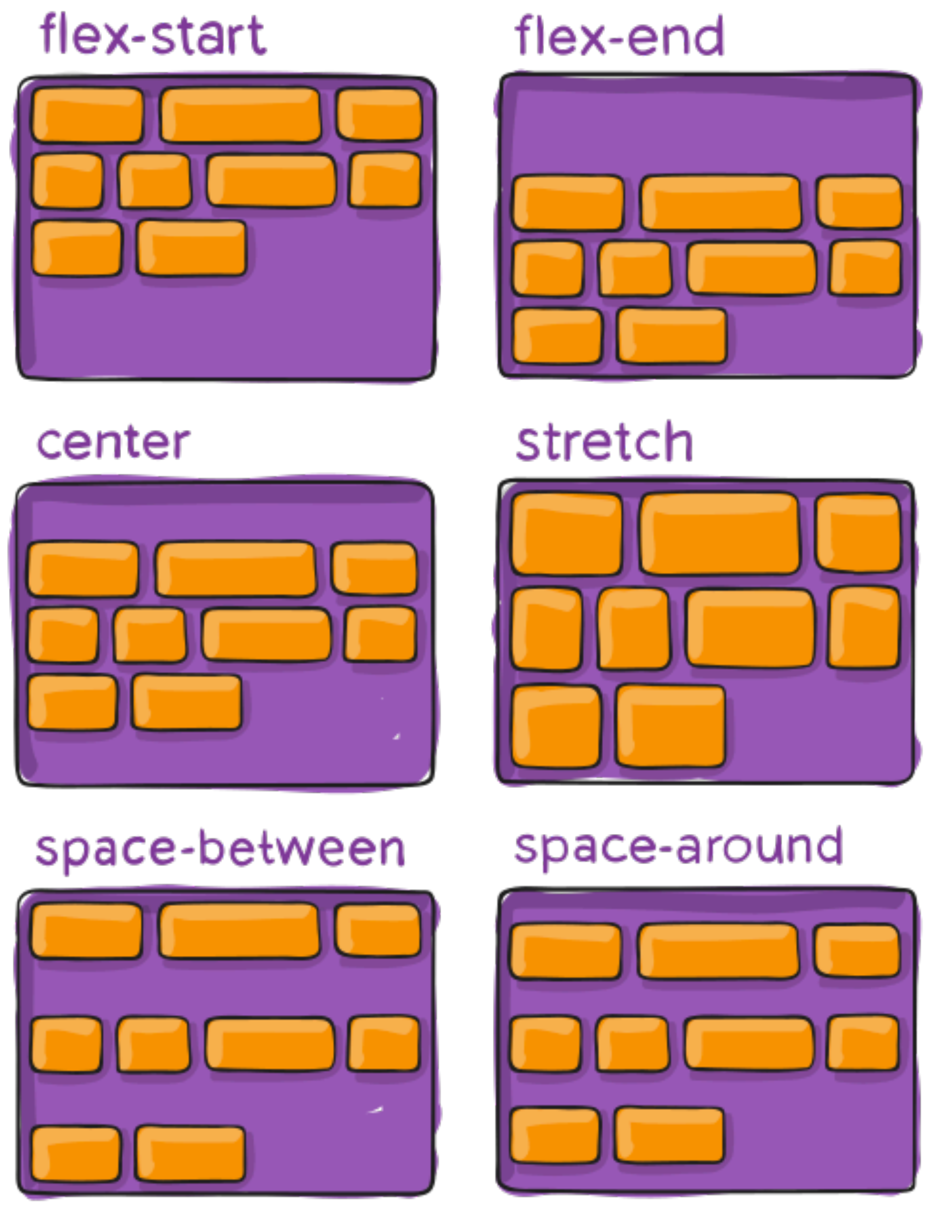
```
.container {
  justify-content:
    flex-start
    | flex-end
    | center
    | space-between
    | space-around
    | space-evenly;
}
```

ALIGN-ITEMS



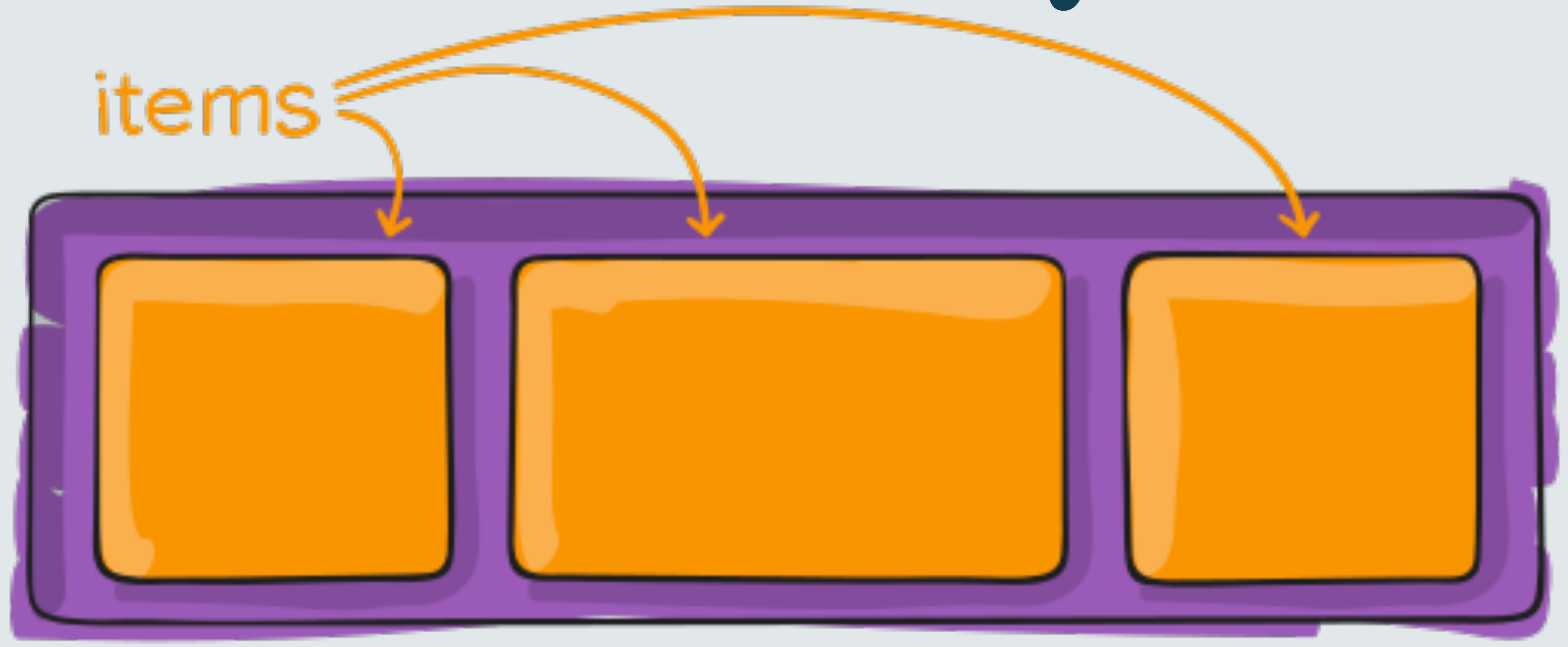
```
.container {
  align-items:
    stretch
     | flex-start
     | flex-end
     | center
     | baseline;
}
```

ALIGN-CONTENT

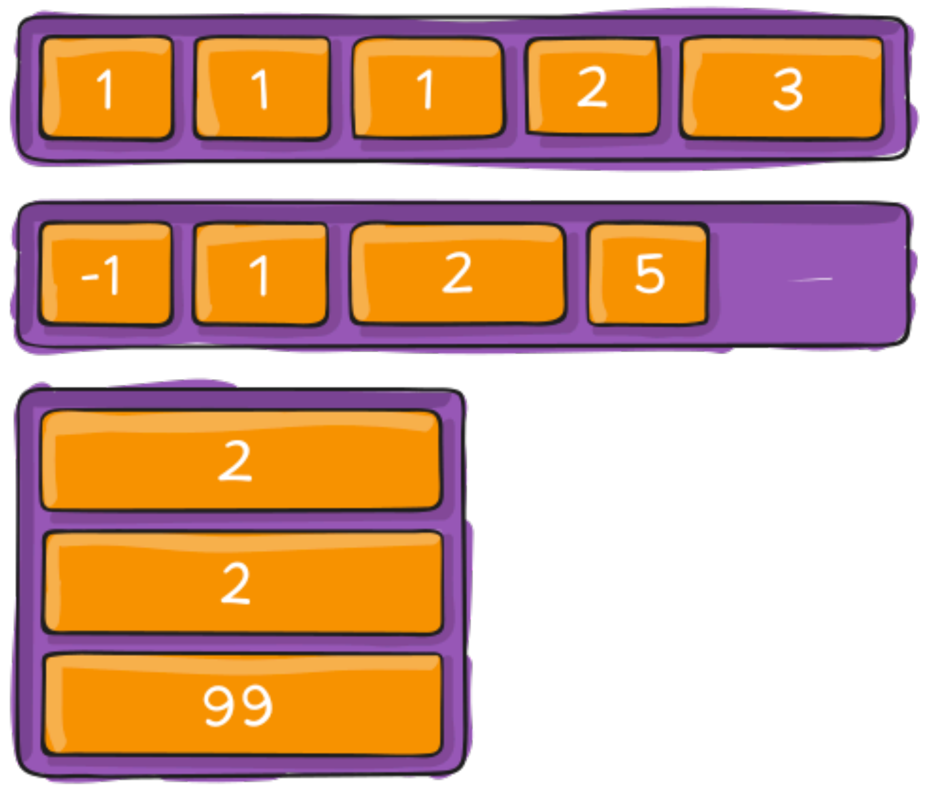


```css
.container {
  align-content:
    flex-start
    | flex-end
    | center
    | stretch
    | space-between
    | space-around;
}
```

Item/Child Properties

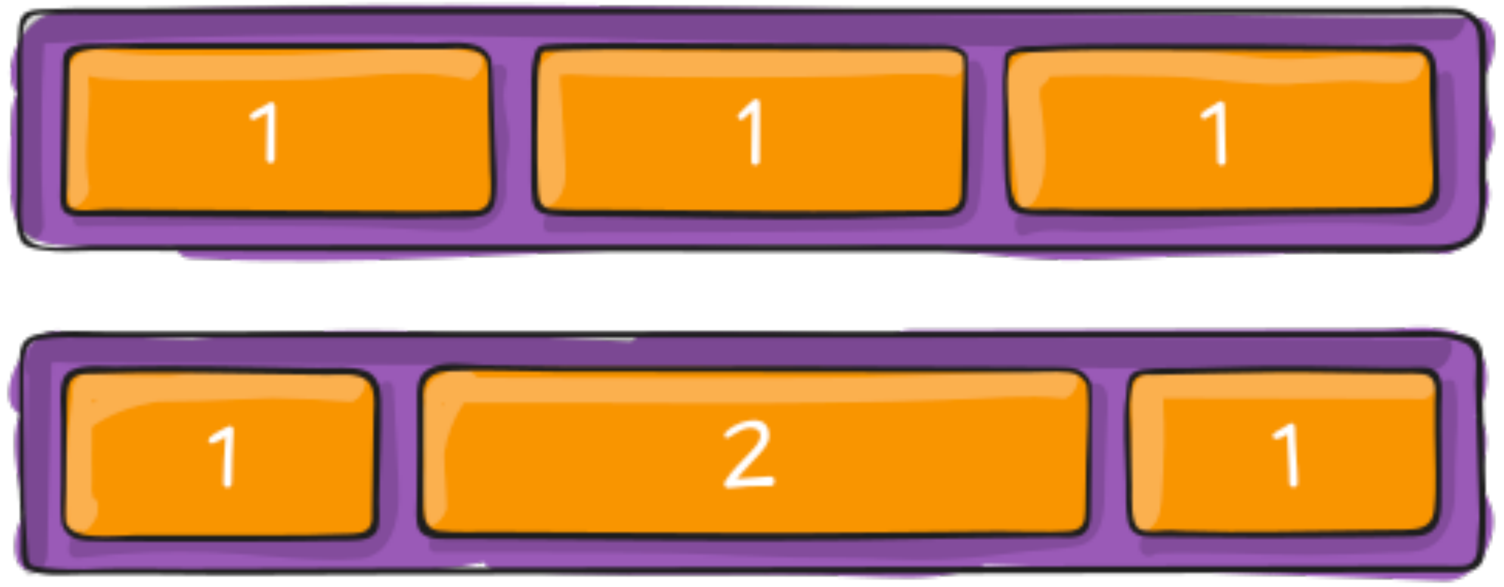


ORDER



```
.item {
  order: <integer>;
}
```

FLEX-GROW/SHRINK/BASIS

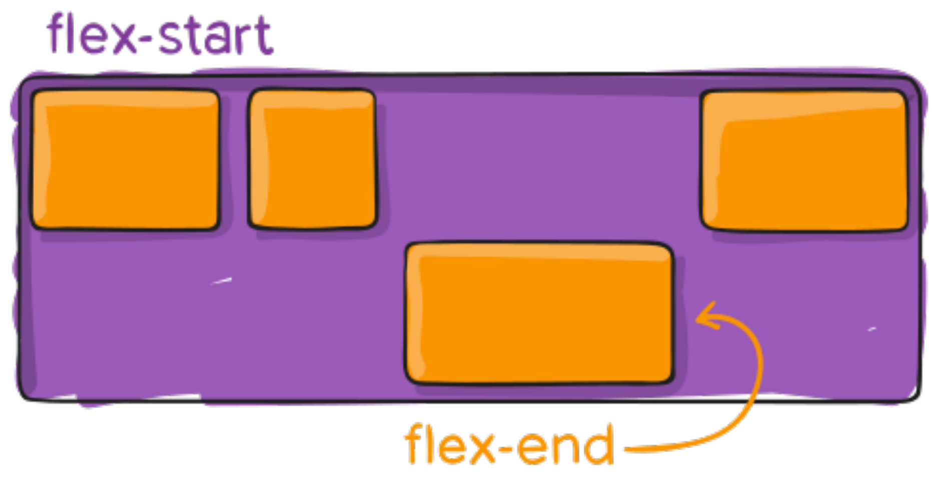


```
.item {
  flex-grow: <integer>;
  flex-shrink: <integer>;
  flex-basis: <length> | auto;
}
```

FLEX

```css
.item {
  flex: none | [ <flex-grow> <flex-shrink>? || <flex-basis> ];
}
```

ALIGN-SELF



```
.item {
  align-self:
    auto
    | flex-start
    | flex-end
    | center
    | baseline
    | stretch;
}
```

RESOURCES

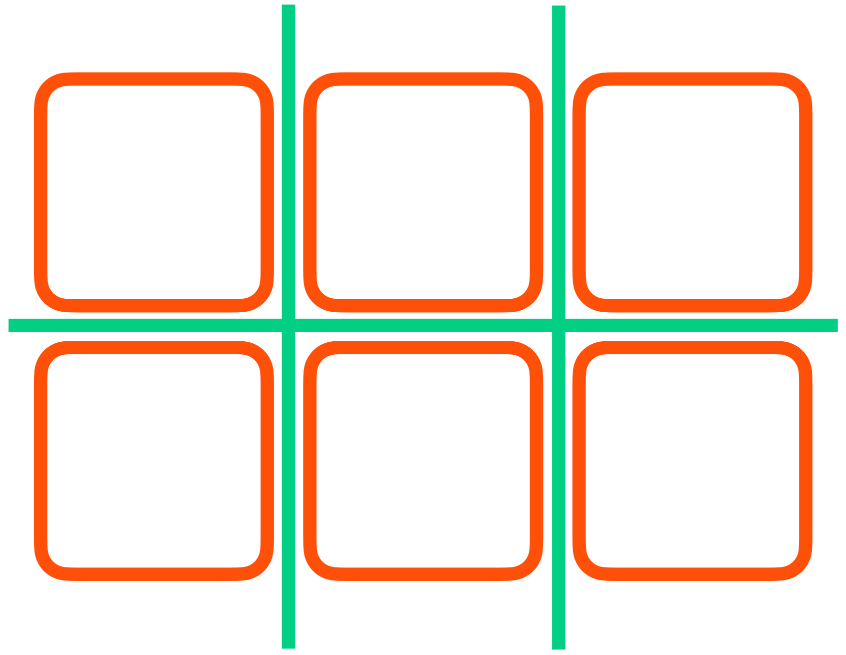
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

https://flexboxfroggy.com/

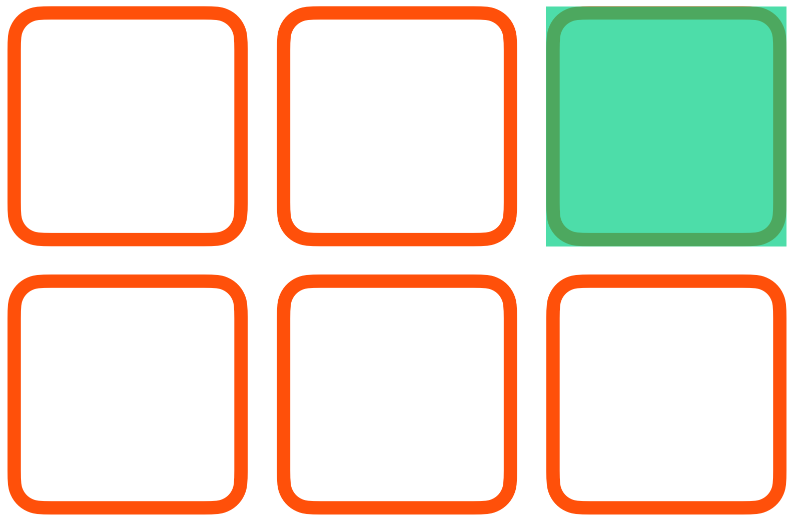
CSS GRID

TERMINOLOGY



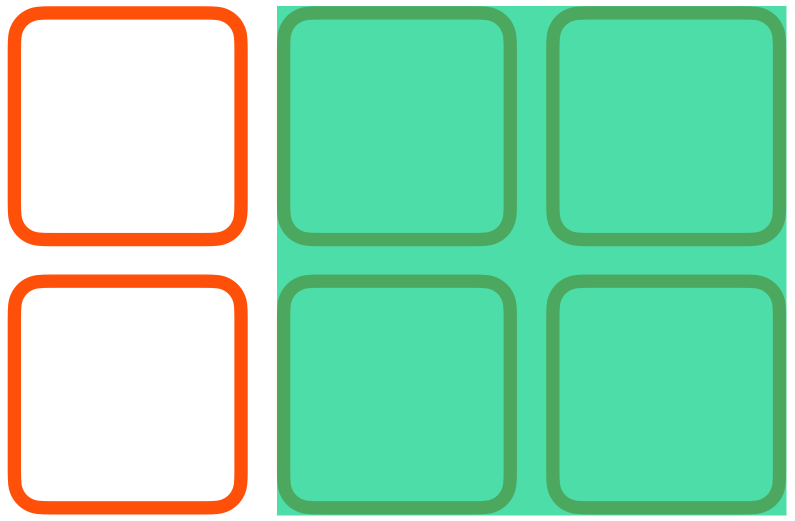
Lines

Vertical and horizontal lines that divide the grid



Cell

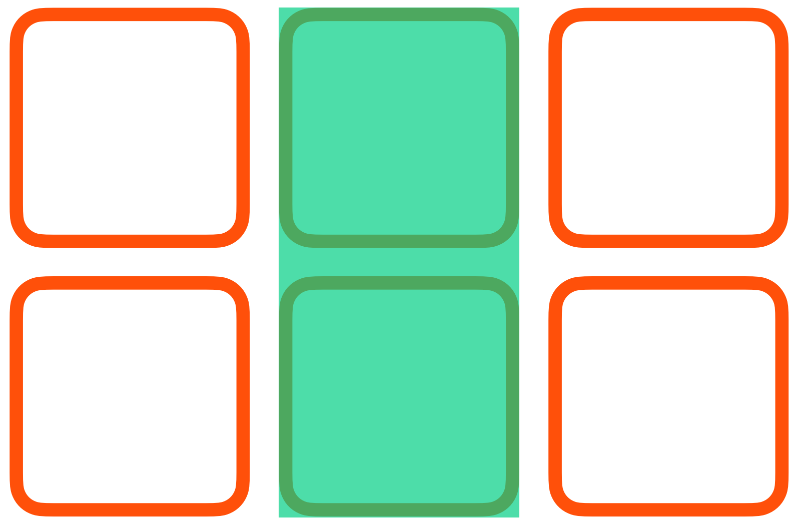
A single unit of a CSS Grid

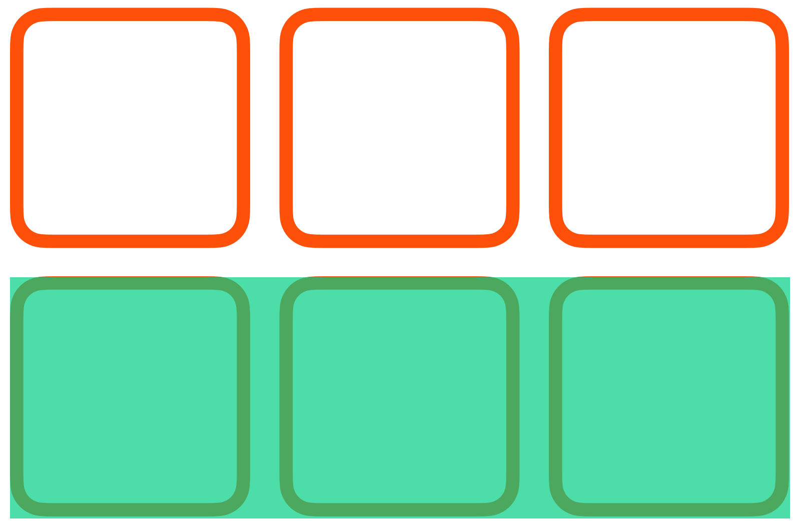


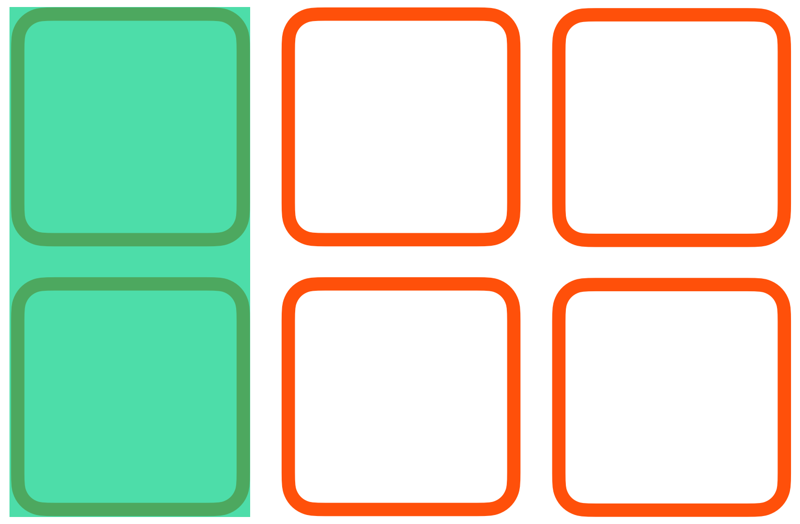
Area

Rectangular space surrounded by four grid lines

TERMINOLOGY







Track

Space between
two grid lines

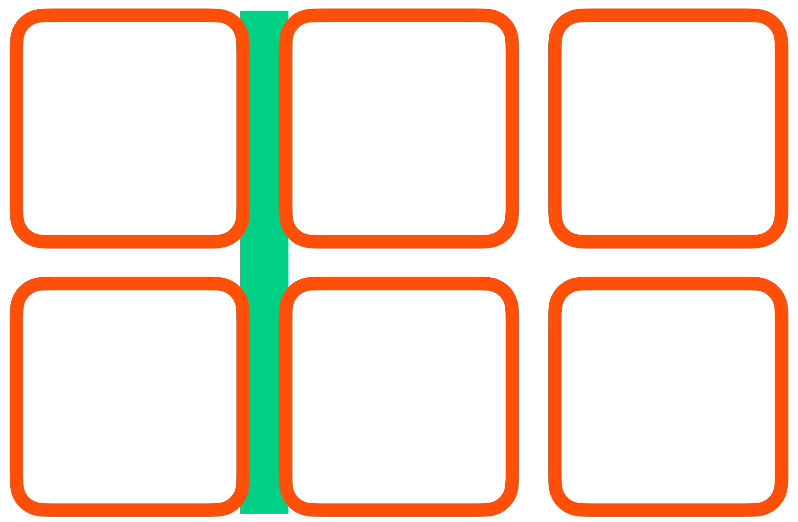
Row

A horizontal track

Column

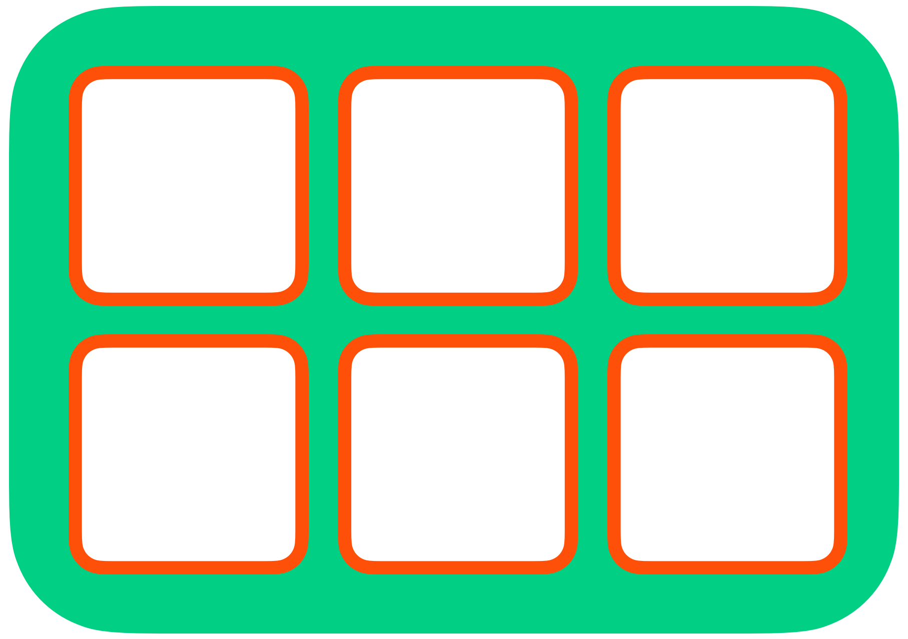
A vertical track

TERMINOLOGY



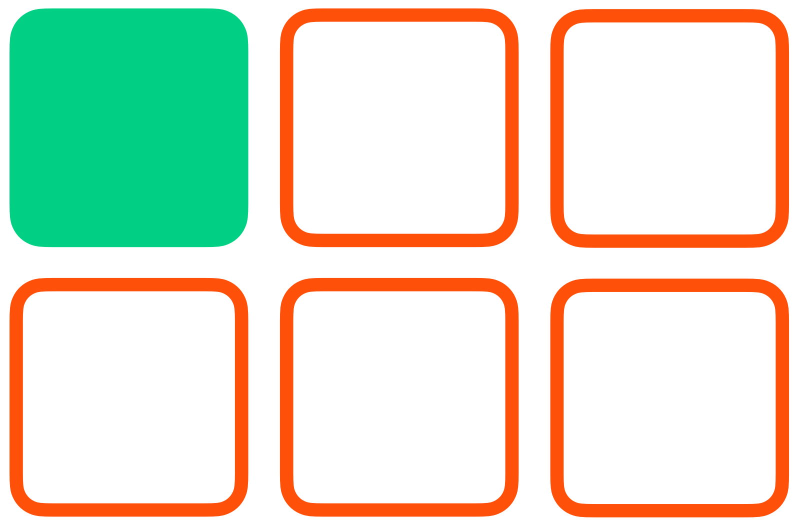
Gutter

Space between
rows and columns



Container

The container that
holds the entire
CSS Grid



Item

Any direct child of
the container

USING GRID

1. Create a grid container:
   `display: grid`

2. Define rows and columns:
   `grid-template-columns` and `grid-template-rows`

3. Add gutter:
   `grid-gap`

HTML

```html
<div class="container">
  <div class="item item1">1</div>
  <div class="item item2">2</div>
  <div class="item item3">3</div>
  <div class="item item4">4</div>
  <div class="item item5">5</div>
  <div class="item item6">6</div>
</div>
```

CSS

```css
.container {
  display: grid;
  grid-template-columns: 150px 150px 150px;
  grid-template-rows: 150px 150px;
  grid-gap: 1rem;
}
```

```css
.item {
  border: 0.25rem solid #FF500A;
  border-radius: 0.5rem;
  display: flex;
  justify-content: center;
  align-items: center;
}
```

CodePen

NEW UNIT: fr

A fraction of available space in the grid container

```
grid-template-columns: 150px 150px 150px;
```

becomes

```
width: calc(450px + 2rem);
grid-template-columns: 1fr 1fr 1fr;
```

what??

calc()

```
width: calc(450px + 2rem);
width: calc(100% - 80px);
width: calc(100% / 6);
font-size: calc(1.5rem + 3vw);
```

Lets you perform calculations when specifying CSS property values

https://caniuse.com/#search=calc https://developer.mozilla.org/en-US/docs/Web/CSS/calc

repeat()

```css
grid-template-columns: 1fr 1fr 1fr;
grid-template-rows: 150px 150px;
```

becomes

```css
grid-template-columns: repeat(3, 1fr);
grid-template-rows: repeat(2, 150px);
```

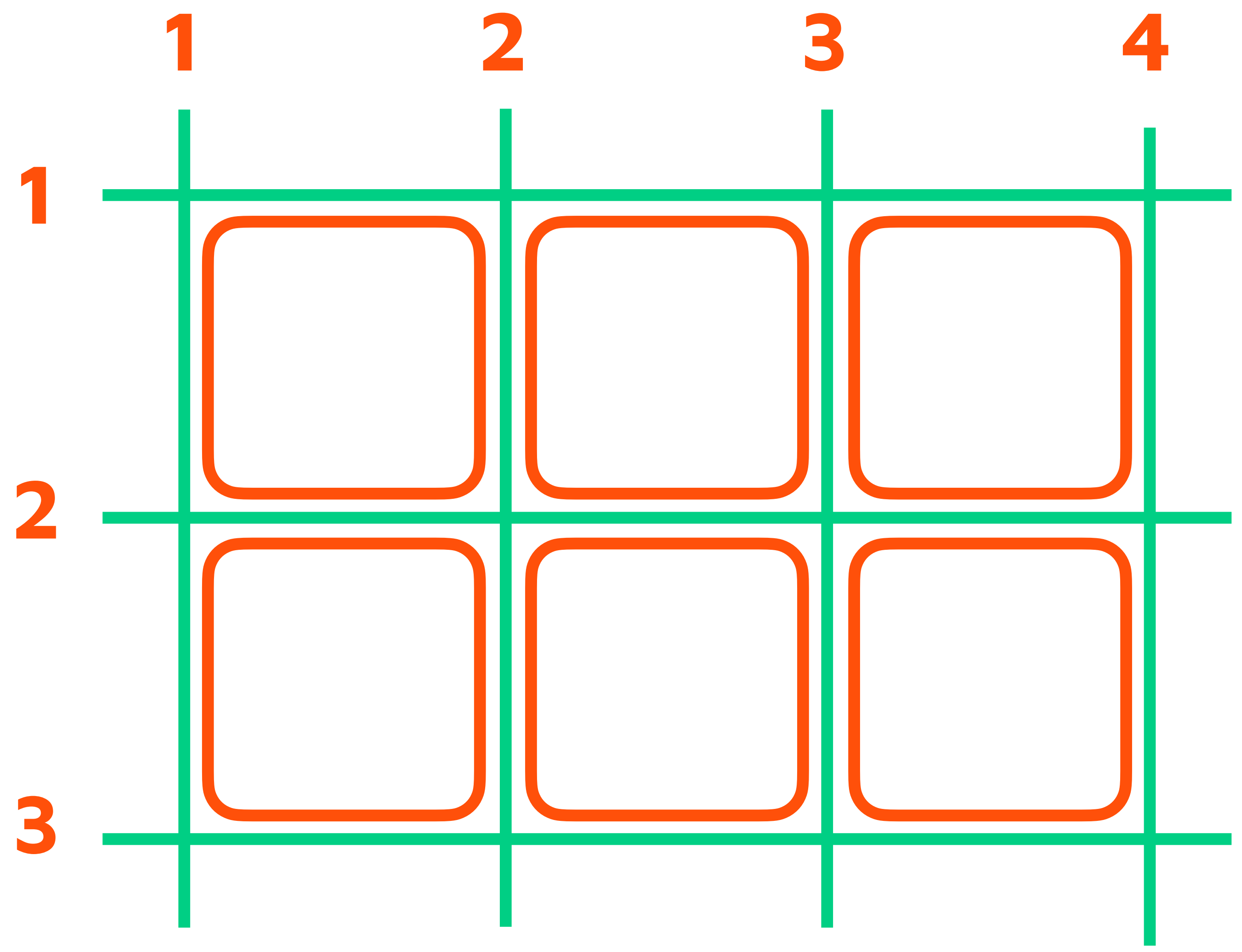
can also be used for part of a listing!

MIXING UNITS

You can mix fixed(px, em) and flexible(%, fr) sizes

```
grid-template-columns: 100px 30% 1fr;
```

POSITIONING ITEMS



```
.item1 {
  grid-row-start: 2;
  grid-row-end: 3;
  grid-column-start: 2;
  grid-column-end: 3;
}
```

or

```
.item1 {
  grid-row: 2 / 3;
  grid-column: 2 / 3;
}
```

Basic Layout

CodePen

TEMPLATE AREAS

1 Add area names to **container**

```
grid-template-areas:
  "header header header"
  "content-1 content-1 sidebar"
  "content-2 content-3 sidebar"
  "footer footer footer";
```

2 Update **item** placement

```
.header {
  grid-row: 1 / 2;
  grid-column: 1 / 4;
}
```

becomes

```
.header {
  grid-area: header;
}
```

NAMED LINES

1 Give names to lines in templates

```
grid-template-columns:
    [main-start content-start] 1fr
    [column3-start] 1fr
    [content-end sidebar-start] 200px
    [sidebar-end main-end];
  grid-template-rows:
    [row1-start] 80px
    [row2-start] 1fr
    [row3-start] 1fr
    [row4-start] 100px
    [row4-end];
```

2 Update **item** placement

```
.header {
  grid-row: 1 / 2;
  grid-column: 1 / 4;
}
```

becomes

```
.header {
  grid-row: row1-start / row2-start;
  grid-column: main-start / main-end;
}
```

RESOURCES

https://mozilladevelopers.github.io/playground/css-grid

https://css-tricks.com/snippets/css/complete-guide-grid/

https://gridbyexample.com/

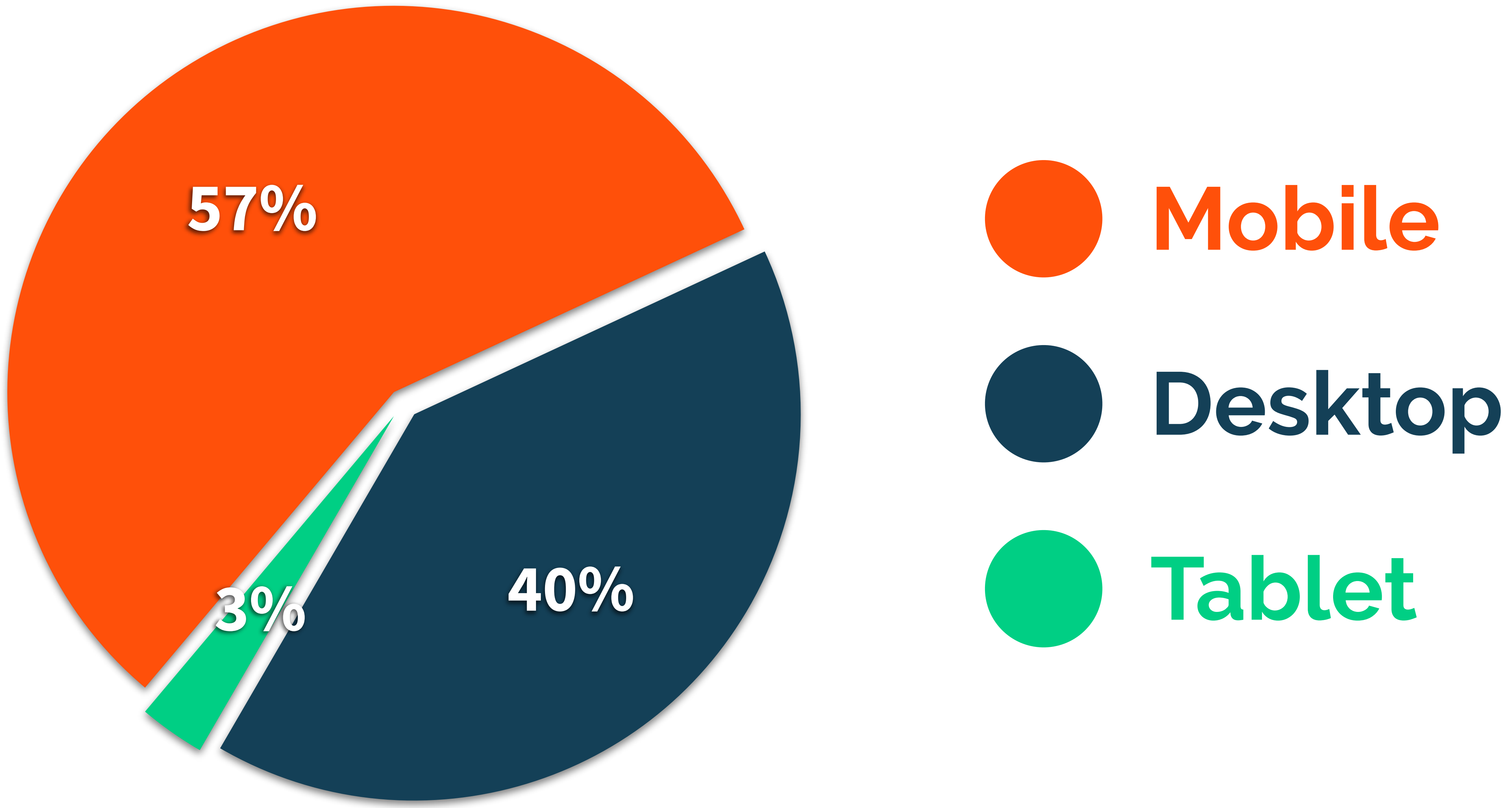
https://cssgridgarden.com/

RESPONSIVE DESIGN

idea

A website should look good and be accessible on every display, from wide screens to mobile devices.

DESKTOP vs MOBILE vs TABLET



https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet

RESPONSIVE DESIGN

Eliminates the distinction between mobile and desktop websites.

Allows a single codebase that is displayed differently in various screen sizes

Achieved through media queries

New York, NY

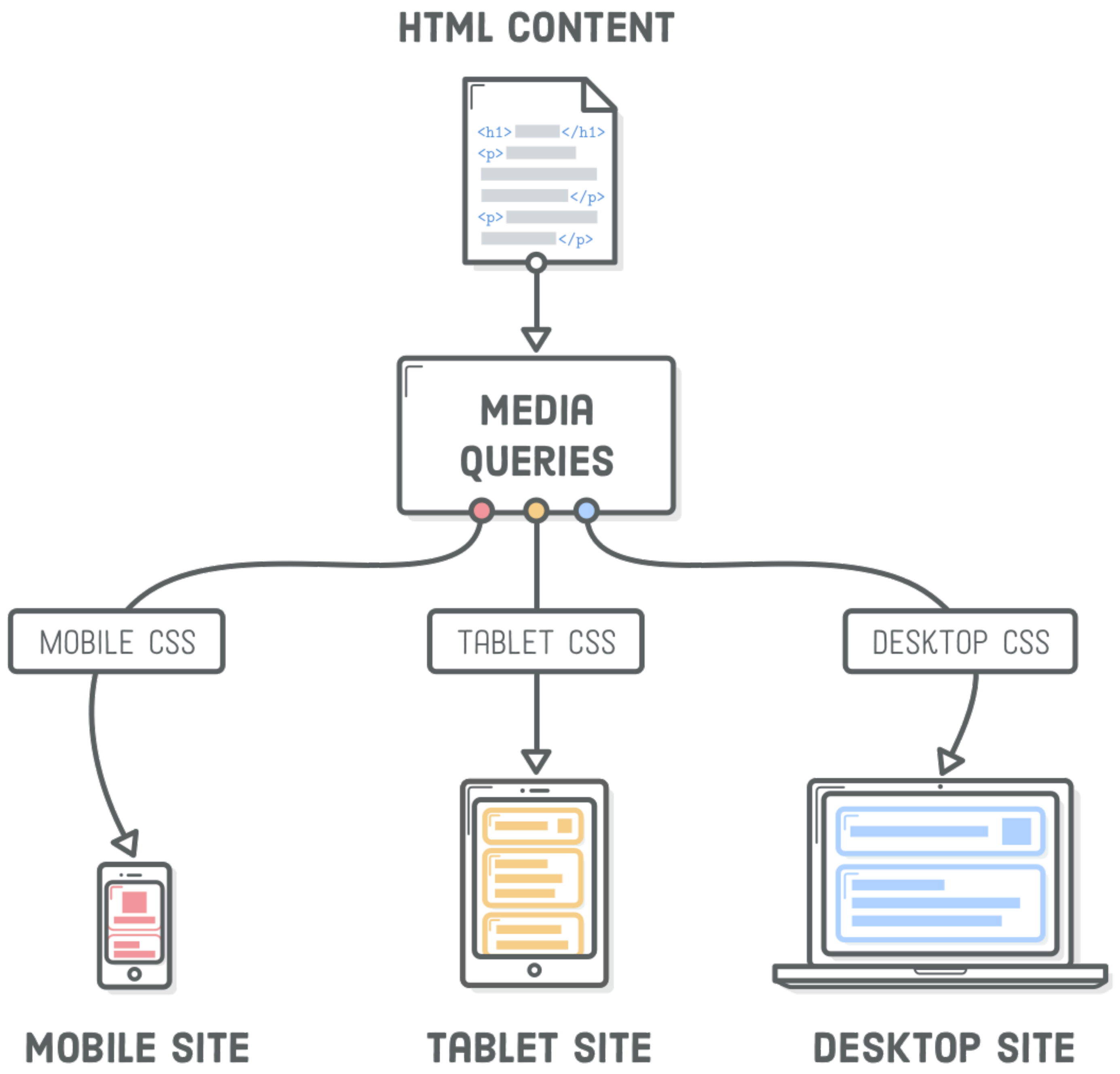
Tuesday, April 15th
Overcast

58 °F

Precipitation: 100%
Humidity: 97%
Wind: 4 mph SW
Pollen Count: 36

Today	68° 36°	Pollen 36
Wednesday	50° 39°	Pollen 36
Thursday	55° 39°	Pollen 36
Friday	54° 43°	Pollen 36

https://developers.google.com/web/fundamentals/design-and-ux/responsive/

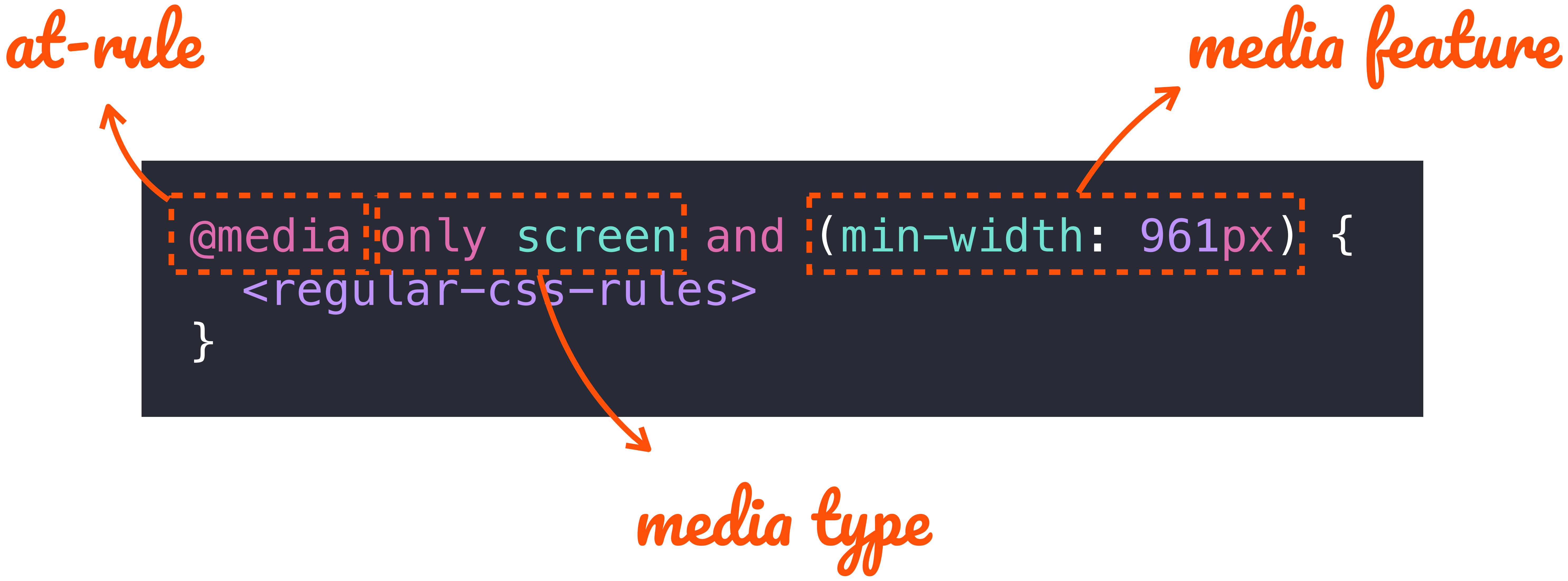


https://www.internetingishard.com/html-and-css/responsive-design/

MEDIA QUERIES

```
@media only screen and (min-width: 961px) {
  <regular-css-rules>
}
```

MEDIA QUERIES



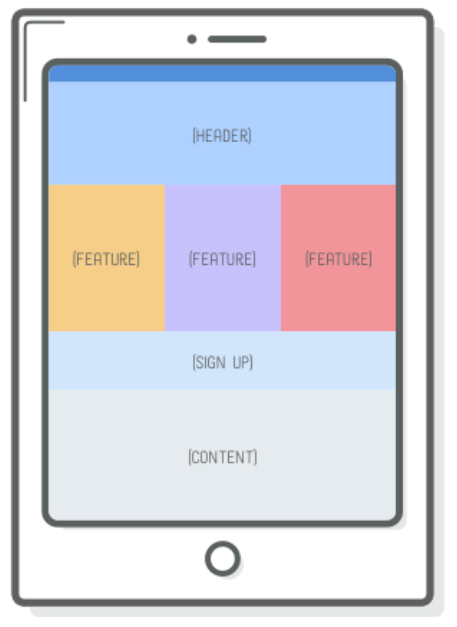
```
@media only screen and (min-width: 961px) {
    <regular-css-rules>
}
```

VIEWPORT ZOOMING

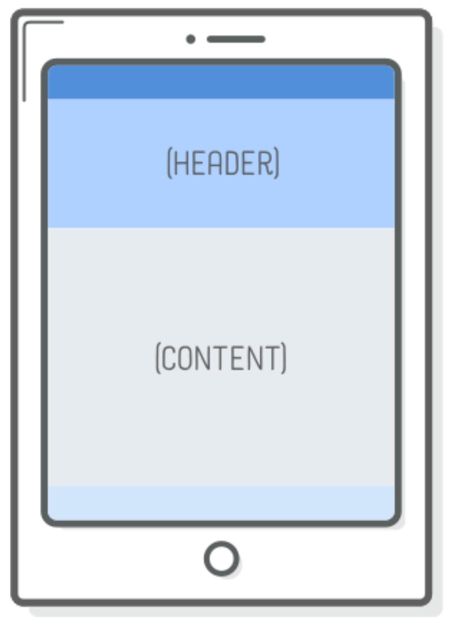
By default, mobile devices zoom out to fit entire desktop layout onto the viewport.

This prevents mobile devices from rendering responsive designs,

In order to disable it, we need to specify a <meta> tag in the <head>



ZOOM ENABLED



ZOOM DISABLED

```
<meta name='viewport' content='width=device-width, initial-scale=1.0'/>
```

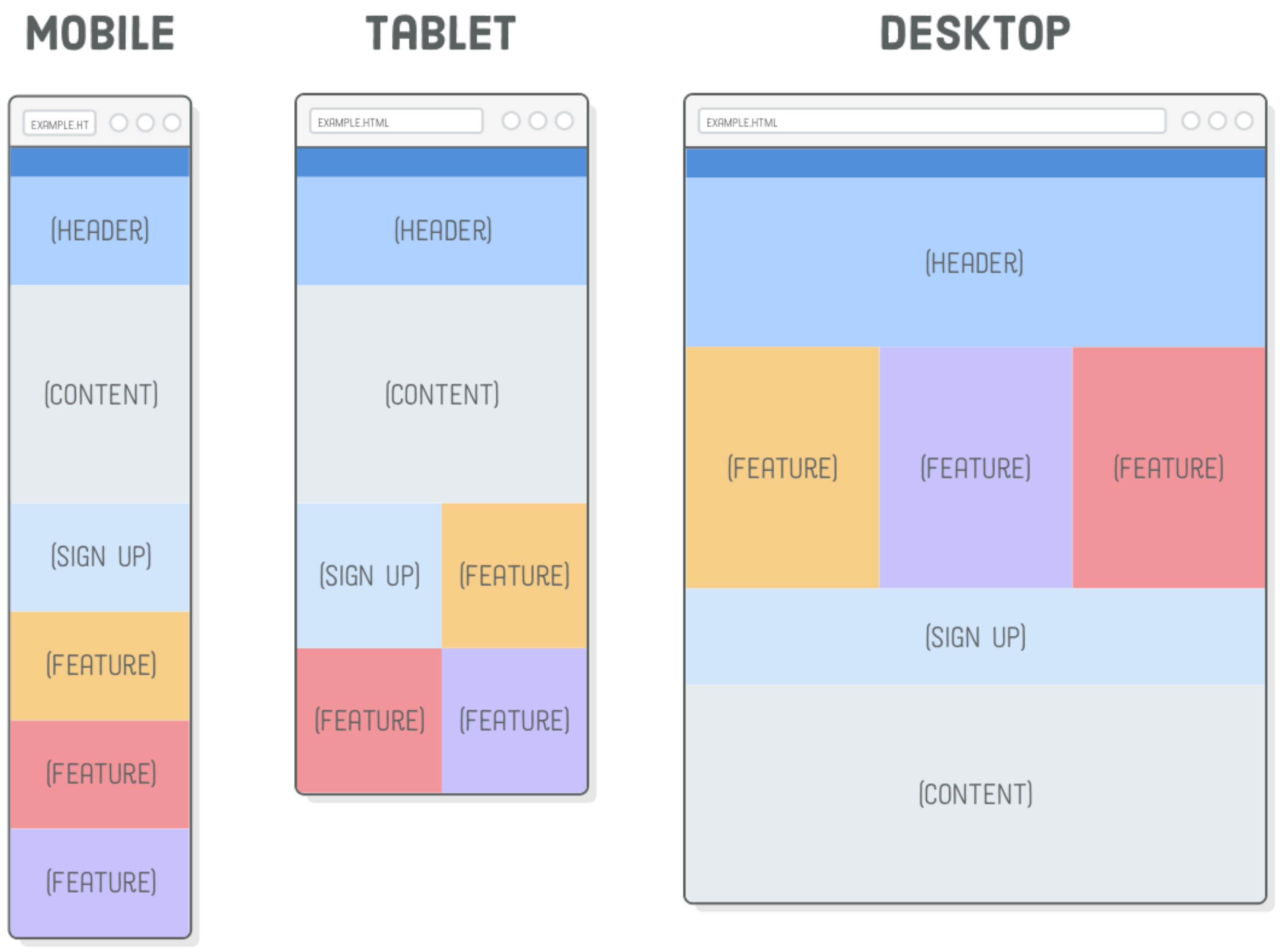
https://www.internetingishard.com/html-and-css/responsive-design/

DESIGN

Start with the design of how the website will look at every breakpoint.

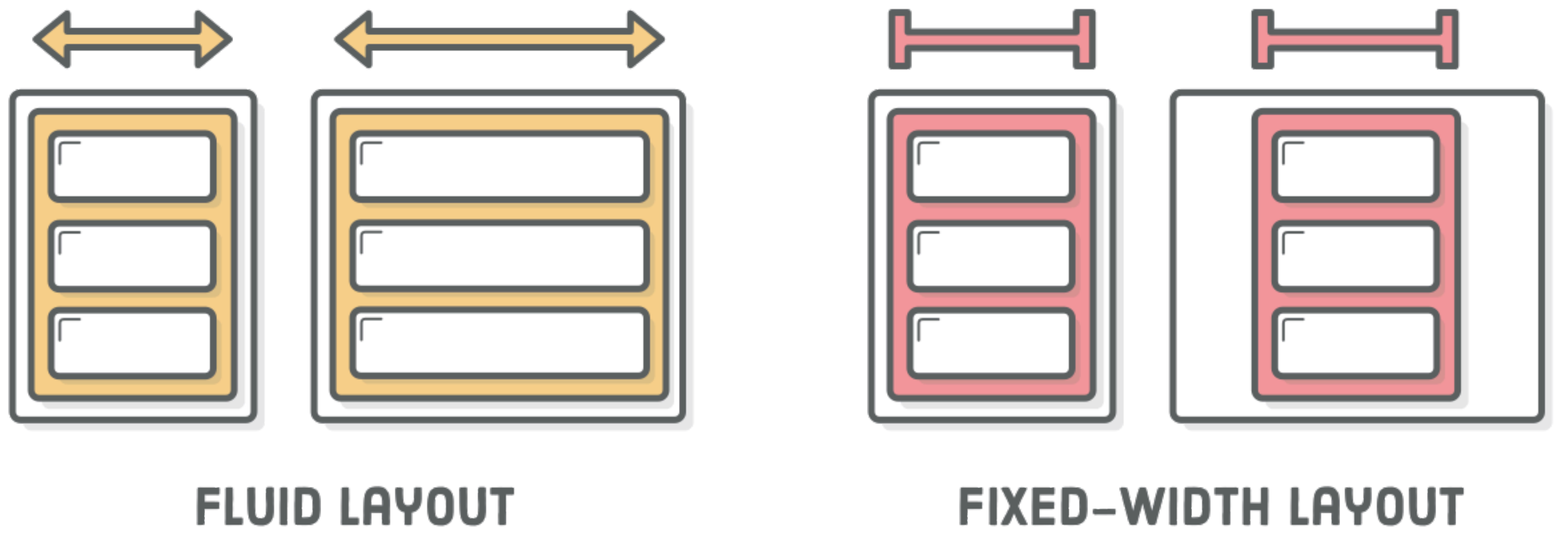
Various responsive design patterns exist. (Mostly Fluid, Column Drop, Layout Shifter, etc.)

Implement them using media queries.



https://www.internetingishard.com/html-and-css/responsive-design/

DESIGN



Fluid Layout: Content stretches/shrinks to fill the entire viewport.

Fixed-Width Layout: Content has the same width regardless of the viewport.

Mobile/Tablet → Fluid   Desktop → Fixed-Width

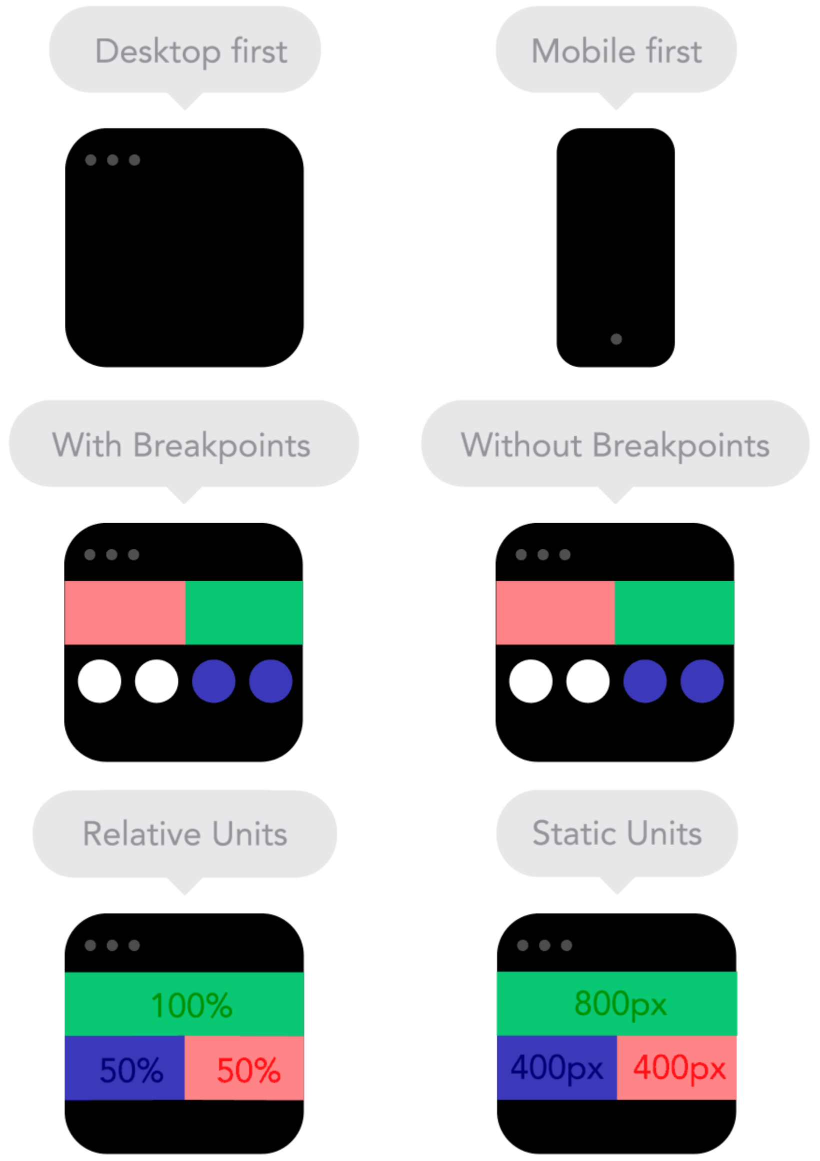
https://www.internetingishard.com/html-and-css/responsive-design/

BASIC PRINCIPLES

Mobile-First vs Desktop-First: Start implementing from one end to maximize code reuse. Usually Mobile-first is more convenient as mobile screens are more restricted.

Choosing Breakpoints: Don't need to be device specific (i.e. iPhone 12 vs Galaxy S21). Take advantage of ranges.

Relative vs Static Units: Use relative units when you want your content to adapt (when you don't have enough screen real-estate), static units when you want the same look (when you have enough space).



https://blog.froont.com/9-basic-principles-of-responsive-web-design/

demo

https://gitlab.com/uiuc-web-programming/responsive-demo

RESOURCES

https://www.internetingishard.com/html-and-css/responsive-design/

https://developers.google.com/web/fundamentals/design-and-ux/responsive/

https://blog.froont.com/9-basic-principles-of-responsive-web-design/

https://alistapart.com/article/responsive-web-design/

NEXT CLASS: JAVASCRIPT

https://uiuc-web-programming.gitlab.io/fa21/