LINKÖPING UNIVERSITY

729G87 Interaction Programming

Lecture 1b – Intro Web Technologies (HTML/CSS)

Philipp Hock, PhD
philipp.hock@liu.se

LINKÖPING UNIVERSITY

CSS

Introduction to Cascading Stylesheets

A quick demo

https://www.csszengarden.com/

Lecture Overview

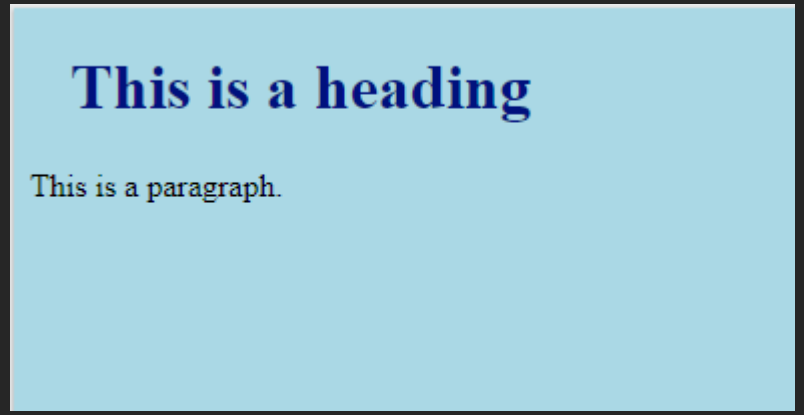
- CSS syntax: selectors, properties and values
- Using classes and id:s
- CSS Box model
- CSS Layout
  - legacy methods vs new methods
  - display models
  - positioning
  - floats
  - flexbox

Use CSS in your HTML file

- External
- Internal
- Inline

External CSS

```html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



"mystyle.css"

```css
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

How to add CSS (w3schools.com)

Internal CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

How to add CSS (w3schools.com)

Inline CSS

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

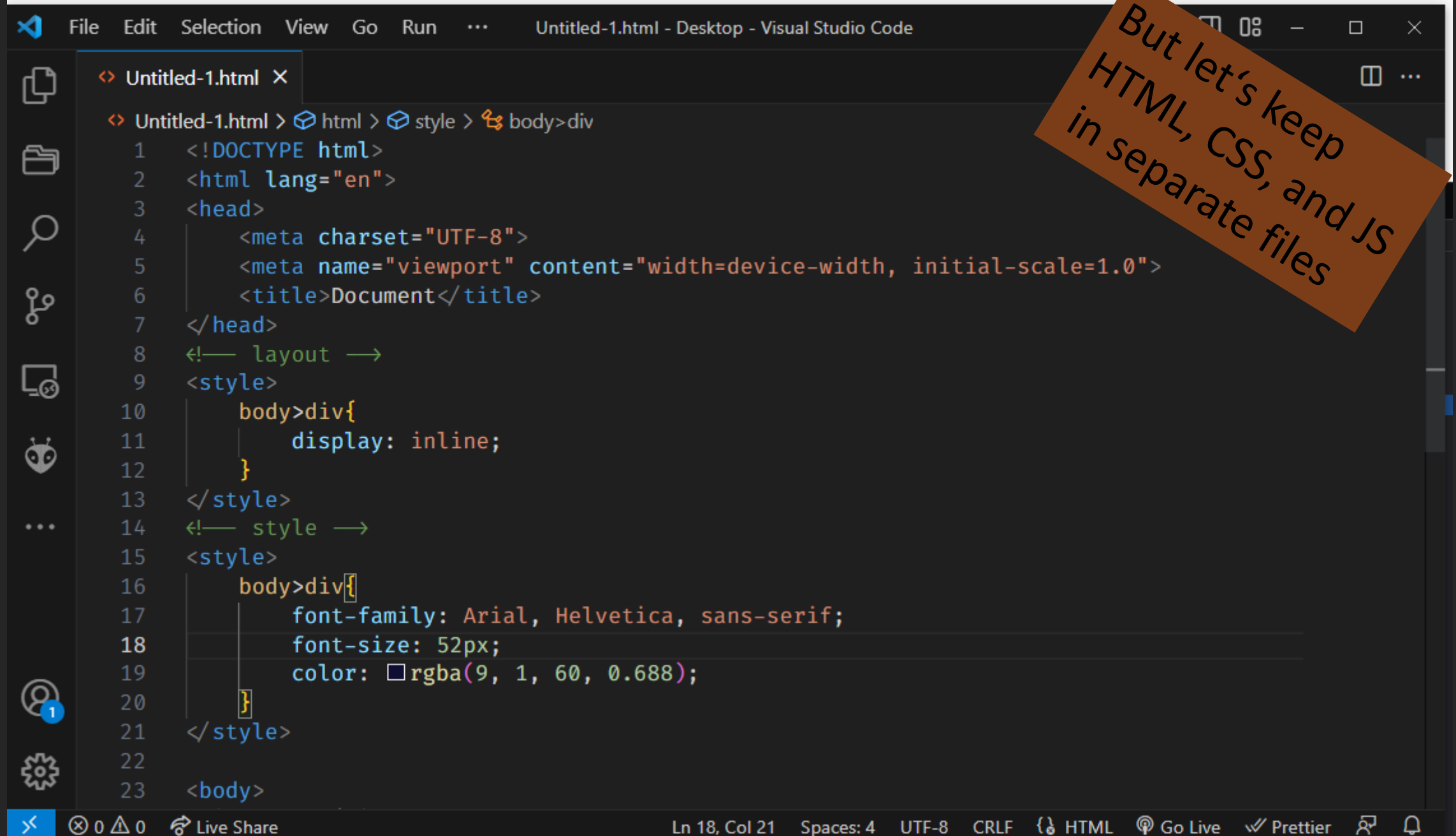
How to add CSS (w3schools.com)

Important facts

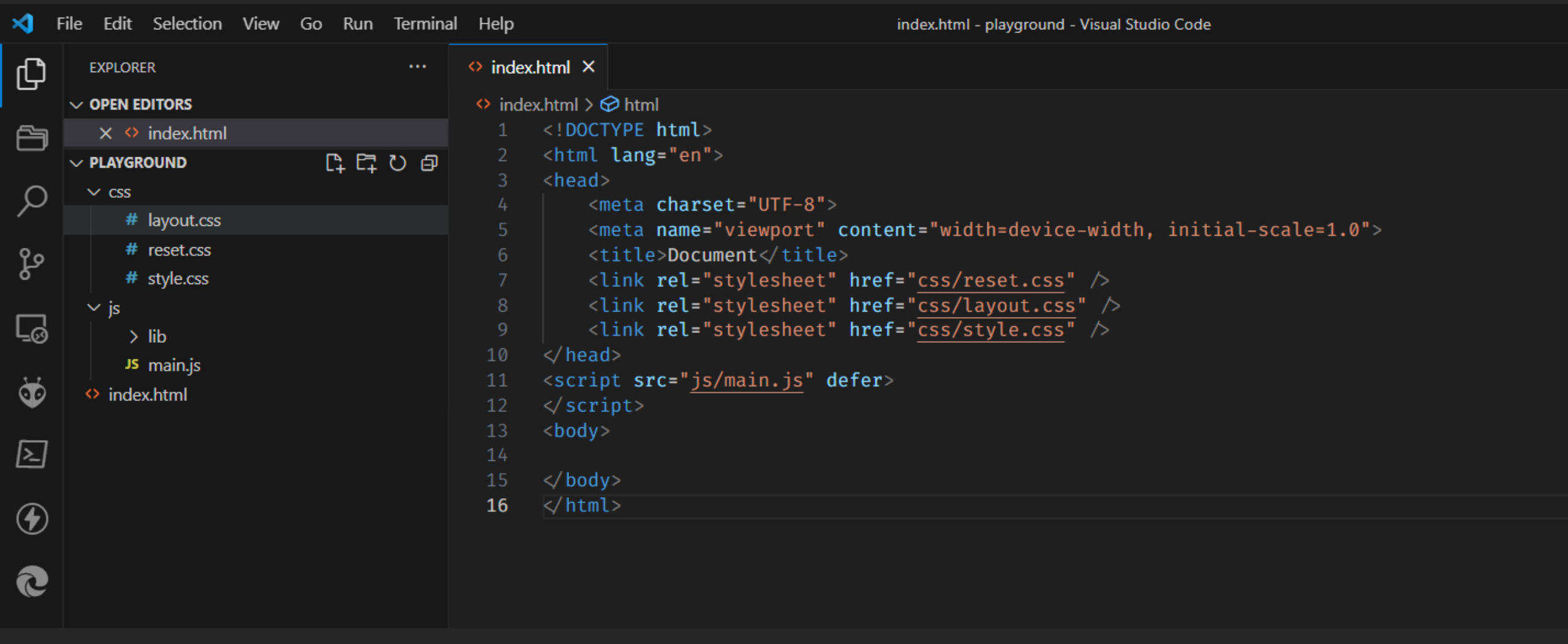
- CSS applies visual style
- CSS applies specific layout infos
- Several ways to achieve things
  - Historically grown
  - Multiple best practices
    - It could be useful to use separate files for layout and style

- HTML elements inherit CSS properties
  - Setting font for <html> tag applies font to all ancestors
  - Rules can be overwritten
  - The most specific rule applies

Hello World


File Edit Selection View Go Run ···
Untitled-1.html - Desktop - Visual Studio Code
Untitled-1.html
Untitled-1.html > html > style > body>div
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Document</title>
7 </head>
8 <!-- layout -->
9 <style>
10 body>div{
11 display: inline;
12 }
13 </style>
14 <!-- style -->
15 <style>
16 body>div{
17 font-family: Arial, Helvetica, sans-serif;
18 font-size: 52px;
19 color: rgba(9, 1, 60, 0.688);
20 }
21 </style>
22
23 <body>
But let's keep HTML, CSS, and JS in separate files
Ln 18, Col 21 Spaces: 4 UTF-8 CRLF HTML Go Live Prettier
Live Share

A barebone template



```css
/* http://meyerweb.com/eric/tools/css/reset/
   v2.0 | 20110126
   License: none (public domain)
*/

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  font: inherit;
  vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
  display: block;
}
body {
  line-height: 1;
}
ol, ul {
  list-style: none;
}
blockquote, q {
  quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
  content: '';
  content: none;
}
table {
  border-collapse: collapse;
  border-spacing: 0;
}
```

css


```css
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
  display: block;
}
body {
  line-height: 1;
}
ol, ul {
  list-style: none;
}
blockquote, q {
  quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
  content: '';
  content: none;
}
table {
  border-collapse: collapse;
  border-spacing: 0;
}
```

Relative vs absolute paths

```
<!-- this file is on https://www.example.org/subfolder/index.html -->

<link rel="stylesheet" href="style.css">
<!-- refers to https://www.example.org/subfolder/style.css -->

<link rel="stylesheet" href="./style.css">
<!-- refers to https://www.example.org/subfolder/style.css -->

<link rel="stylesheet" href="/style.css">
<!-- refers to https://www.example.org/style.css -->

<link rel="stylesheet" href="../style.css">
<!-- refers to https://www.example.org/style.css -->

<link rel="stylesheet" href="./../style.css">
<!-- refers to https://www.example.org/style.css -->
```

./ means current directory
../ means go up one directory

Implicit rules

- Attributes sometimes have no effects
  - Mostly depending on display and position attribute
  - This makes CSS so confusing!
- Common attributes are
  - width
  - height
  - padding
  - margin
  - position
  - z-index
  - display
- Mind the *display* attribute, for example
  - *Width* and *height* apply on display: block, but not on inline elements.
  - More ‘‘hidden“ rules
  - CSS is easy to learn, hard to master

CSS Units

https://wpengine.com/resources/choose-css-unit-create-better-site-layouts-how-to/

https://developer.mozilla.org/en-US/docs/Web/CSS/length

- Absolute units
  - px (pixels) px (not a pixel on real screen, it accomodates pixel density as well)
    - http://inamidst.com/stuff/notes/csspx
  - in (inches)
  - cm (centimeter)
  - mm (millimeter)
  - pc (picas)
  - pt (points)

- Font relative units
  - em (1em = inherited font-size, 2em = double inherited font-size)
  - rem (1rem = font-size of root elemen)
  - ex
  - ch

- Viewport relative units
  - vh (viewport height) (1vw = 1% of width of the viewport's initial containing block (i.e. the <html> block in most cases))
  - vw (viewport width) (1vh = 1% of height of the viewport's initial containing block)
  - vmin (viewport minimum) (1% of the smallest viewport unit, either vh or vw)
  - vmax (viewport maximum) (1% of the largest viewport unit, either vh or vw)

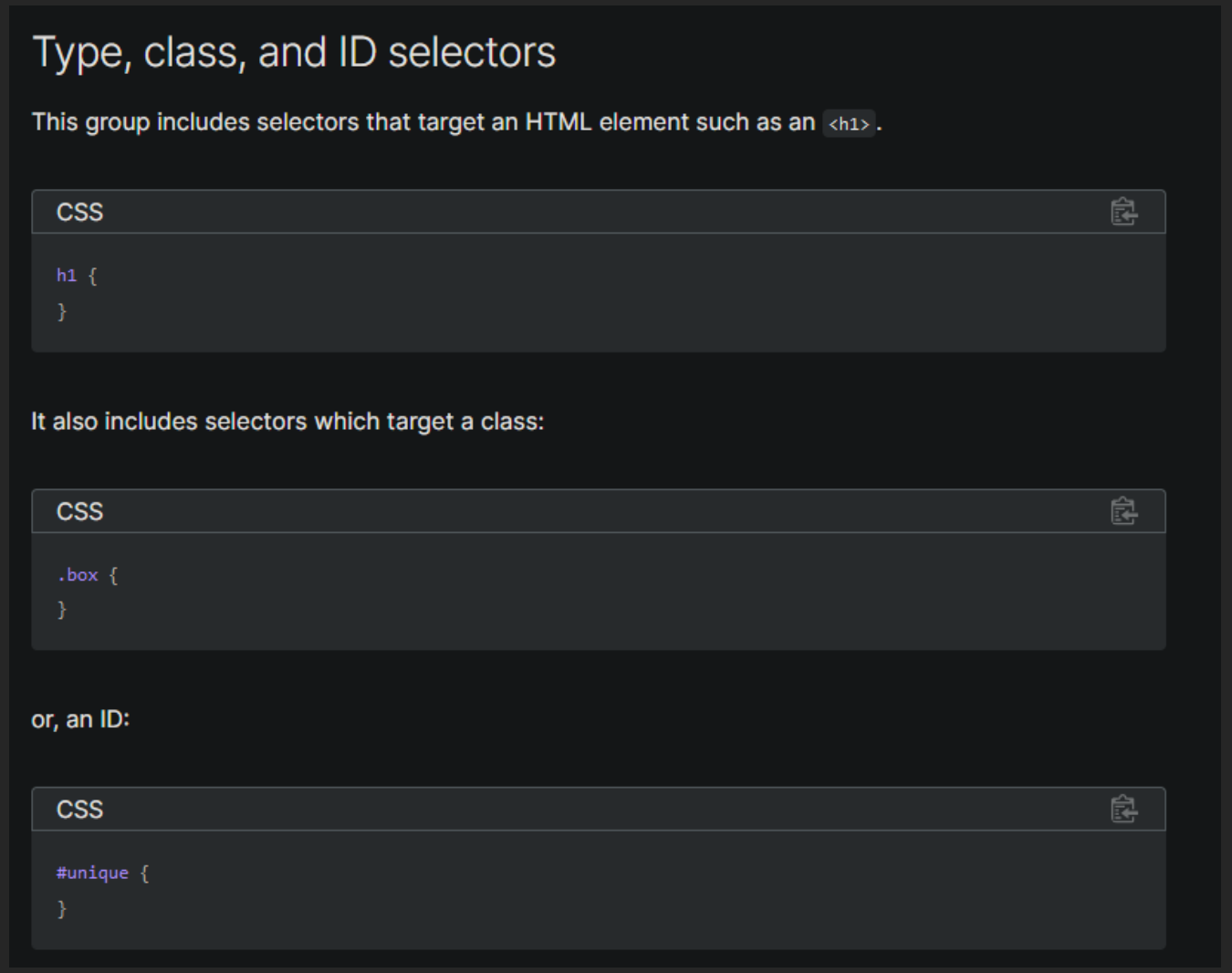
CSS Selectors

https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Selectors

```css
/* id == "a" and id == "b" */
#a, #b {
    background-color: red;
}

/* id == "d" that is a child of id == "c" */
#c #d {
    background-color: blue;
}

/* id == "f" that is a direct child of id == "e" */
#e>#f{
    background-color: white;
}
```

CSS Selectors

https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Selectors



Type, class, and ID selectors

This group includes selectors that target an HTML element such as an `<h1>`.

CSS

```
h1 {
}
```

It also includes selectors which target a class:

CSS

```
.box {
}
```

or, an ID:

CSS

```
#unique {
}
```

CSS Selectors

https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Selectors

Attribute selectors

This group of selectors gives you different ways to select elements based on the presence of a certain attribute on an element:

CSS

```
a[title] {
}
```

Or even make a selection based on the presence of an attribute with a particular value:

CSS

```
a[href="https://example.com"]
{
}
```

CSS Selectors

https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Selectors

Pseudo-classes and pseudo-elements

This group of selectors includes pseudo-classes, which style certain states of an element. The `:hover` pseudo-class for example selects an element only when it is being hovered over by the mouse pointer:

CSS

```
a:hover {
}
```

It also includes pseudo-elements, which select a certain part of an element rather than the element itself. For example, `::first-line` always selects the first line of text inside an element (a `<p>` in the below case), acting as if a `<span>` was wrapped around the first formatted line and then selected.

CSS

```
p::first-line {
}
```

CSS Selectors

https://www.w3schools.com/cssref/css_selectors.php

CSS vars

```
:root {
    --half-gutter-width: 32px;
}

.sidebar {
    margin-left: var(--half-gutter-width);
    margin-right: var(--half-gutter-width);
}
```

Also have a look at css functions:
https://www.w3schools.com/cssref/css_functions.php

CSS rules

```css
.nesting {
  color: hotpink;
}

.nesting > .is {
  color: rebeccapurple;
}

.nesting > .is > .awesome {
  color: deeppink;
}
```

```html
<div class="nesting">
  This text will be hotpink.
  <div class="is">
    This text will be rebeccapurple.
    <div class="awesome">
      This text will be deeppink.
    </div>
  </div>
</div>
```

Use of classes and IDs

- Classes are useful to address mutliple elements
- Prefer tags over classes (<nav> better than div class=“nav“)
- Use meaningful names, avoid names that imply styles
  - avoid e.g., class=‘‘bold‘‘
- Classes can be combined with tags: div.myClassName

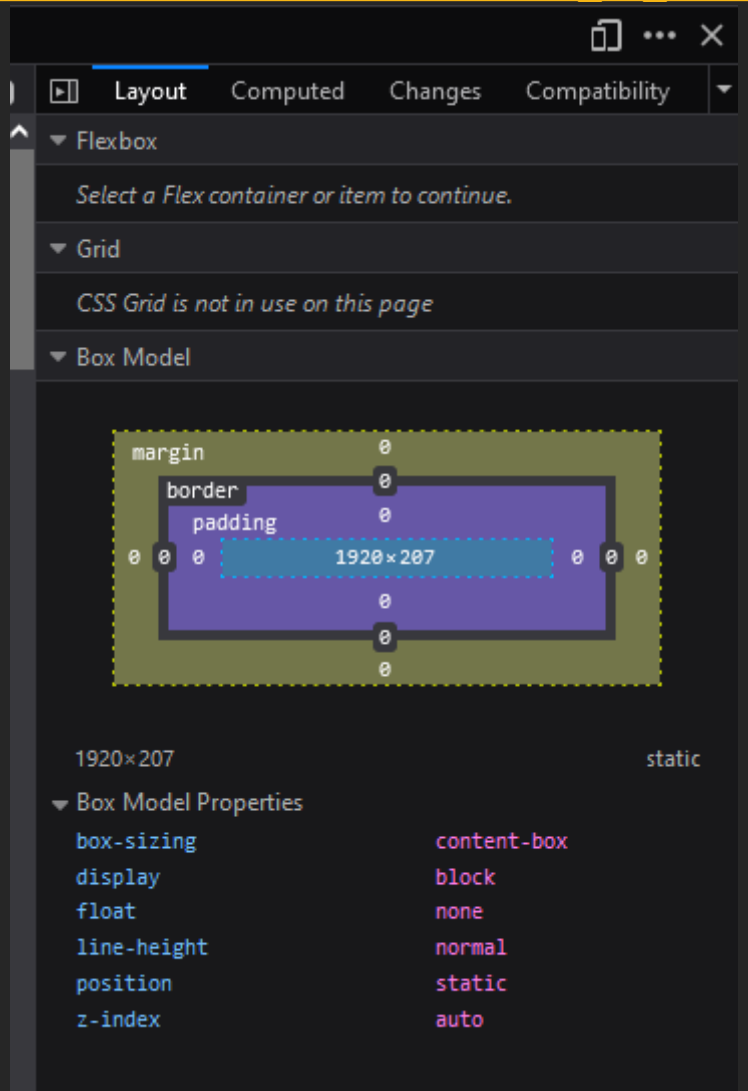
- Use Ids for unique elements
- If every element in your DOM has an ID, you did it wrong
- If your Ids contain numbers, you might prefe: https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-child

Rules of thumb for selectors

- Try to be as unspecific as possible in order to be able to reuse code.
- Less rules can be overridden by more specific rules when needed.
- Avoid relying on IDs (for styling). Try to use classes or paths, as these make it possible to reuse styles, i.e. "Don't repeat yourself." (DRY)

CSS Box Model

https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Box_model



Layouts

- Normal flow (flow layout) + positioning
  - display: block | inline | inline-block
  - position: static | relative | fixed | absolute | sticky
- Floating elements (legacy use for layout)
  - float: left | right | none
- Flexbox layout
  - (parent manages children for a column/row based layout)
  - display: flex
  - This is your friend
- Grid layout (define a parent grid an place children within the grid)
  - display: grid

How to layout a page?

- Break the page down into structures
  - Ignore the inner structures, work from big to small
- Flexbox will solve most of your problems
  - Flexbox can be arranged in rows or columns
    - Think in rows
    - and sometimes also in columns
- Use background-colors to identify your containers
  - Fill with blind text or set min-heights

Inner and outer display types

- Outer display type: how the element will be displayed inside other elements
  - display: block | inline
- Inner display type: how child elements will be displayed
  - display: flex | grid

For a complete list, see

https://developer.mozilla.org/en-US/docs/Web/CSS/display

Positioning

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

- position: static (default)
- position: relative (adjusts the static property)
- position: fixed (relative to the viewport)
- position: absolute (relative to parent)
- position: sticky

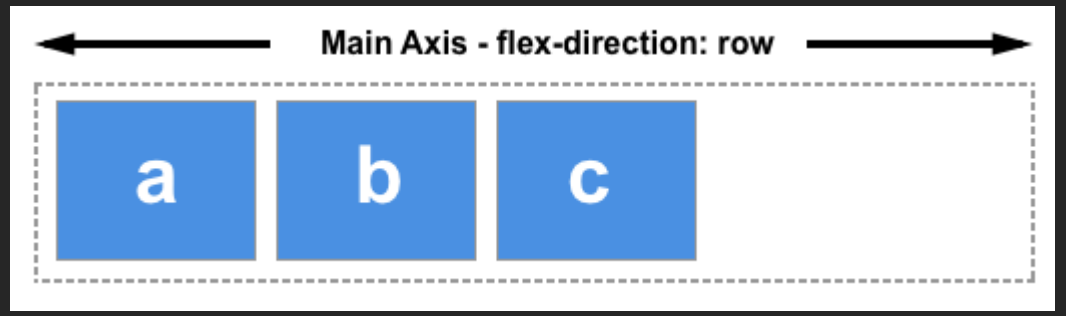
  (sticks to a specific position when moved to that position)

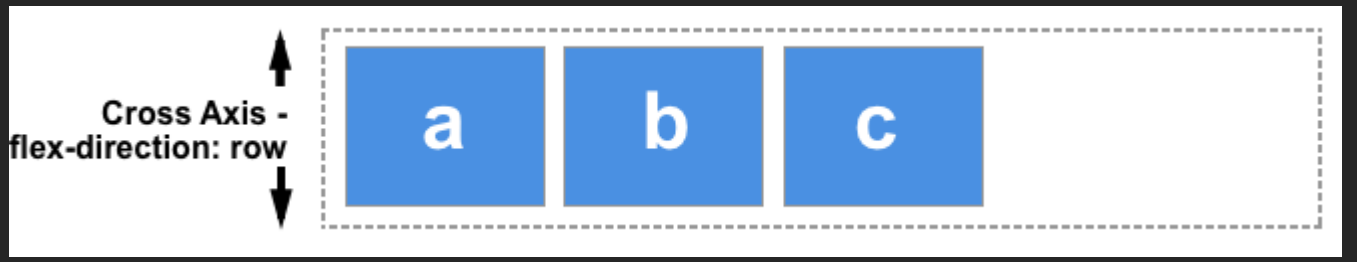
- Play around with positioning to learn & see!

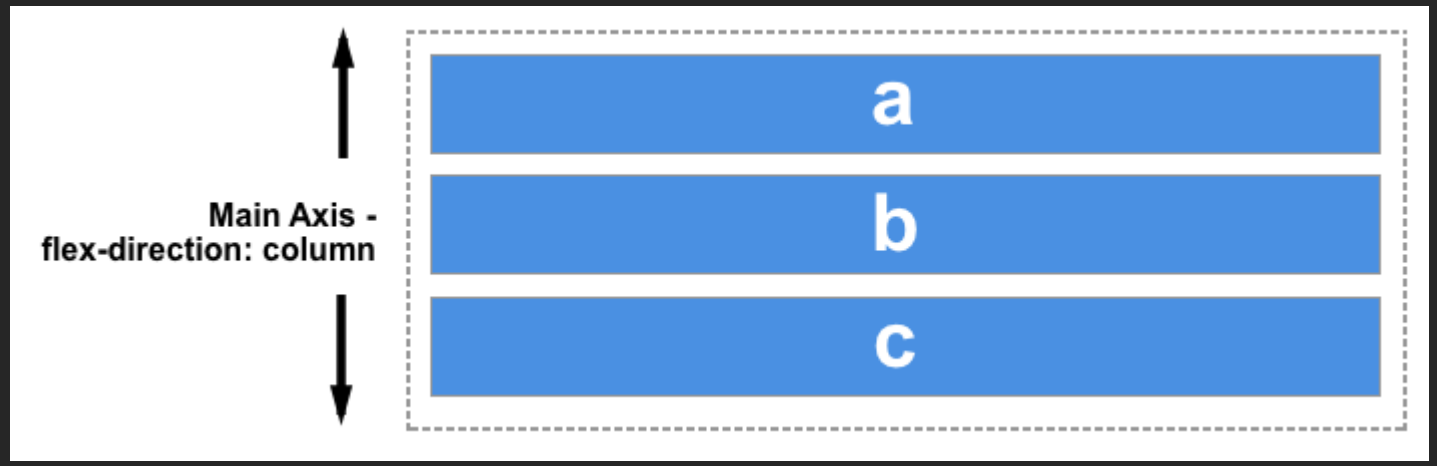
  https://blog.webdevsimplified.com/2022-01/css-position/

Flexbox

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox







Flexbox

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox

- Main-axis & Cross axis
- Layout child elements as rows or columns
- Properties for configuring spacing and size of child elements

- Some properties are set **on** the **container** (also called **parent**, flex container)
- Some properties are set on the elements **in** the **container** (also called **children**, flex items)

Flexbox - Container

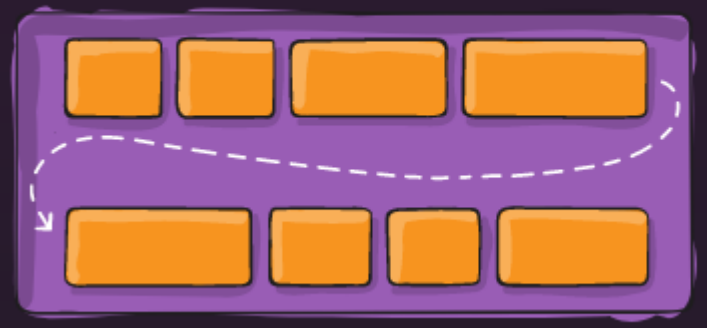
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox

- display: flex (sets the flexbox property)
- flex-direction: row | row-reverse | column | column-reverse
  (sets the main axis direction)
- flex-wrap: nowrap | wrap | wrap-reverse
  (single vs multi row)
- justify-content: flex-start | flex-end | center | space-between
  (controls the alignment of all elements on the main axis)
- align-items: stretch | flex-start | flex-end | center | baseline
  (controls the alignment of all elements on the cross axis)
- align-content: start |center | space-between | space-around
  (distribution of space between and around content along cross-axis, try:
  flex-warp: wrap;)
- flex-flow: <flex-direction> <flex-wrap> (shorthand)

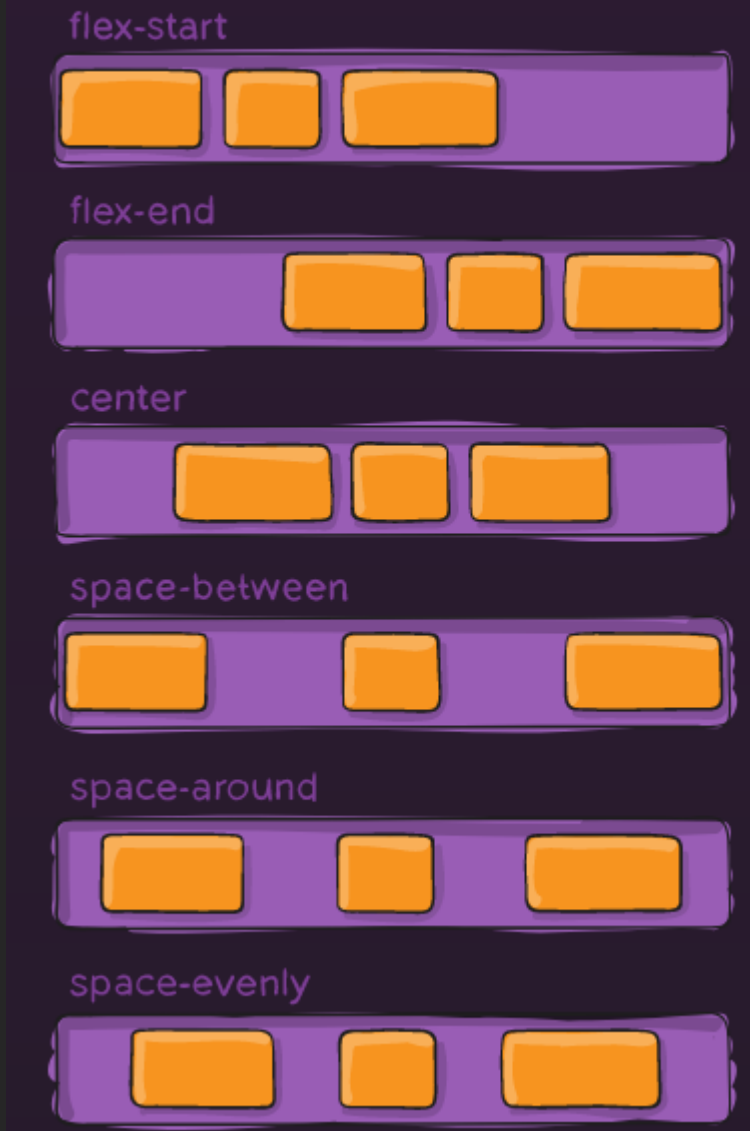
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

flex-direction

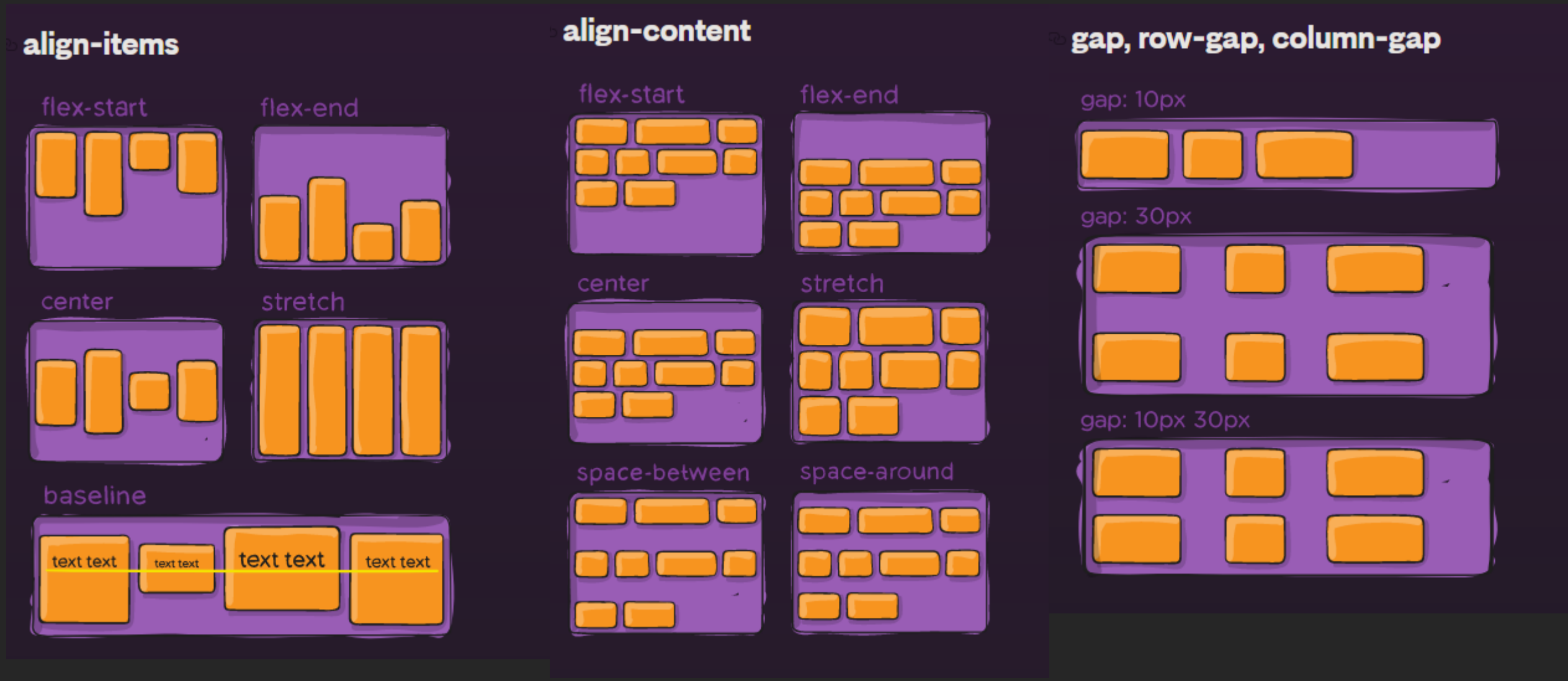
flex-wrap



justify-content



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



Flexbox - Items

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox

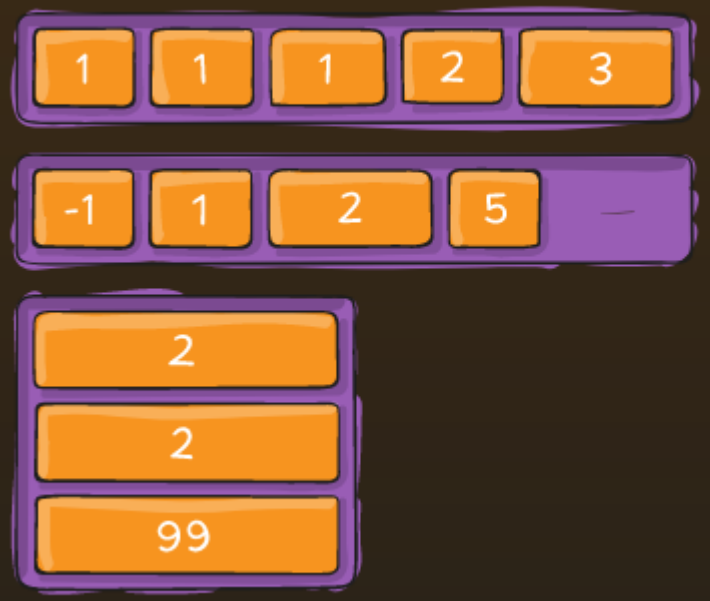
- order: <integer> (reposition elements)
- flex-grow: <positive number> (controls growth by a weight)
- flex-shrink: <positive number> (controls negative growth by a weight)
- flex-basis: <width/height> (set the initial width/height of a flex item)

- align-self: auto | flex-start | flex-end | center | baseline |stretch;
  (set alignment of a single flex item along the cross axis)
- justify-self
  (Set the alignment of a single flex item within its alignment container along the main axis)

Important: Flexbox will override the size property of its children!

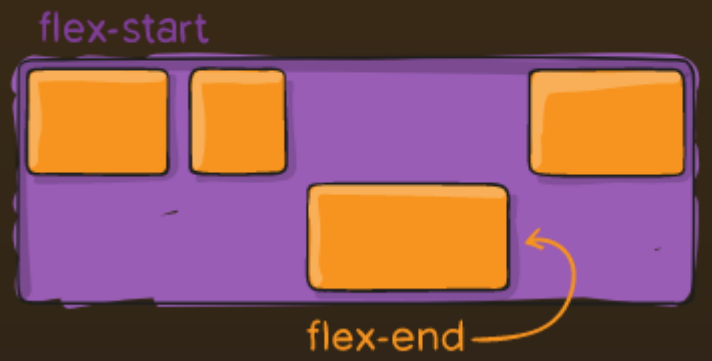
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

order



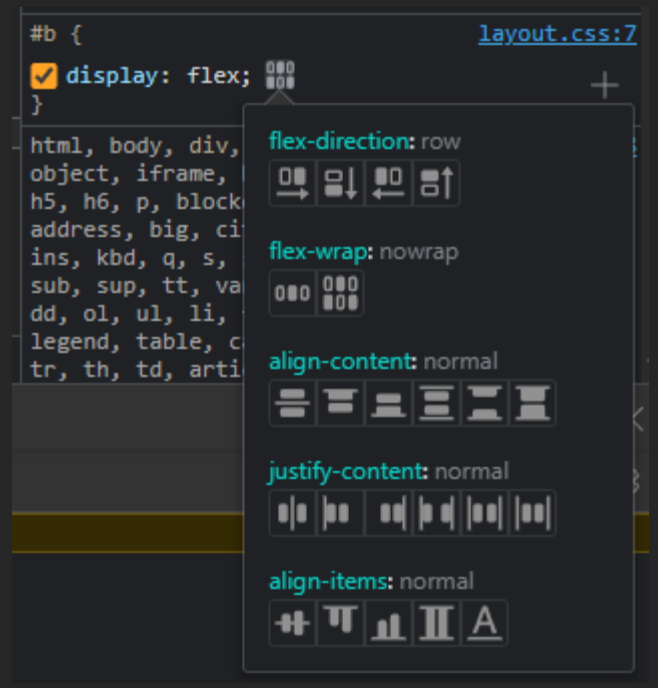
flex-grow

align-self



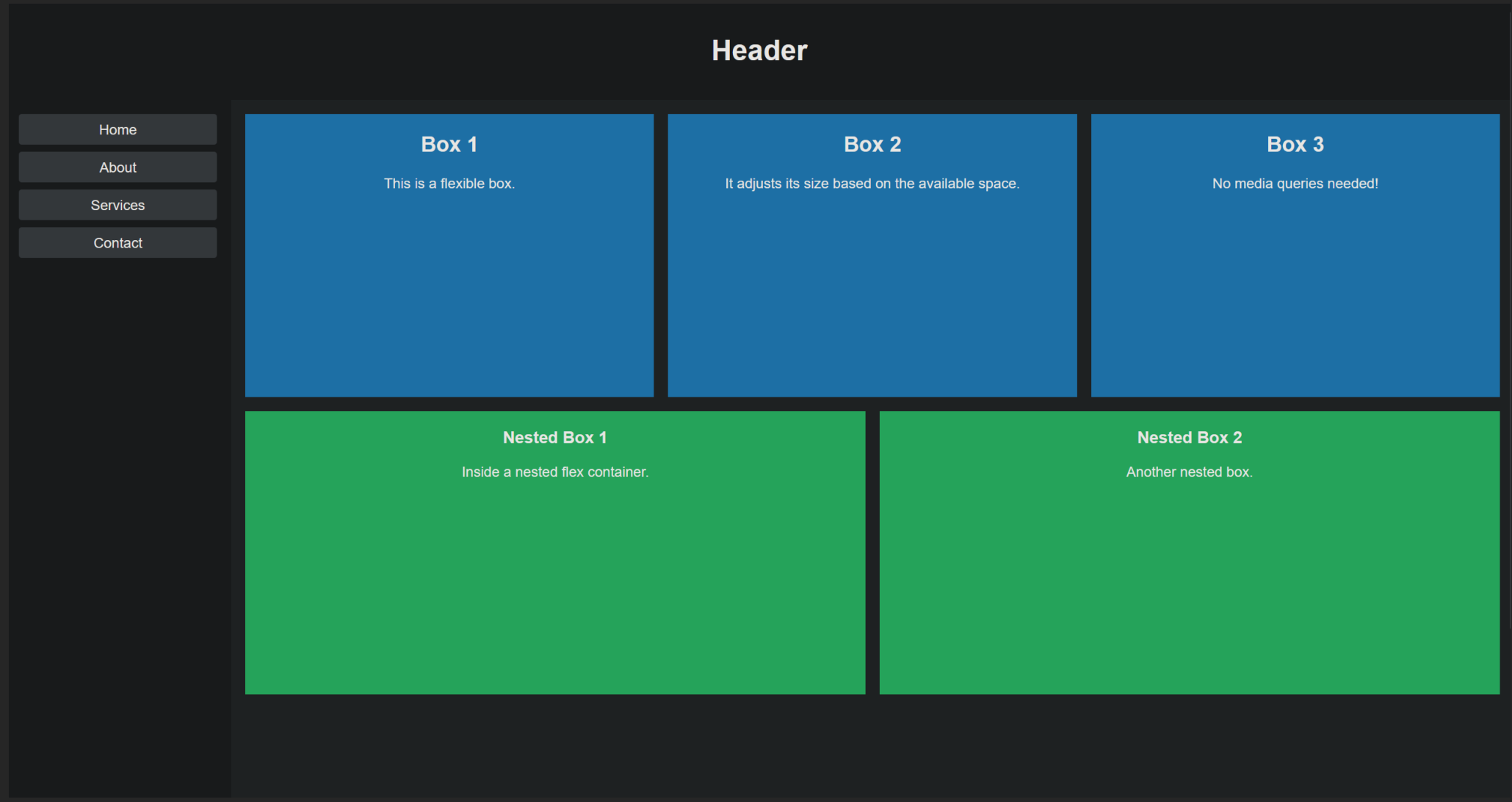
Flexbox

- Intuitively understand flexbox by using chrome‘s dev tools (F12 / ctrl+shift+i)
- Select a flexbox element
- Change the values and see whats happening

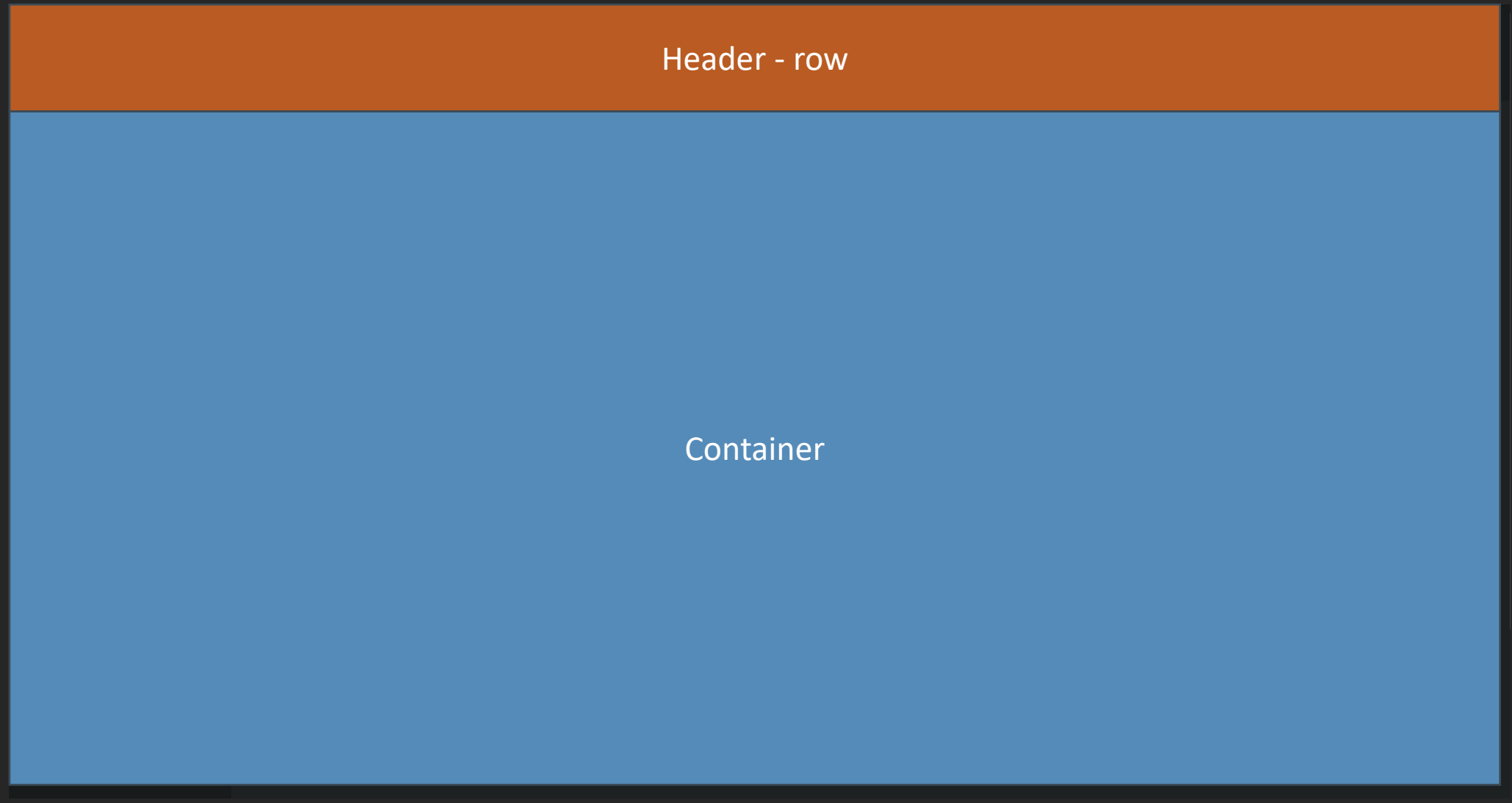


https://codepen.io/Philipp-Hock/pen/rNojQJM

Deconstruct a Layout



Deconstruct a Layout



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
<style>
  body, html{
    margin: 0;
    min-height: 100vh;
    display: flex;
    flex-direction: column;

  }

  #container{
    background-color: blue;
    flex: 1;
}
</style>
</head>

<body>
  <header></header>
  <div id="container">

  </div>

</body>
</html>
```

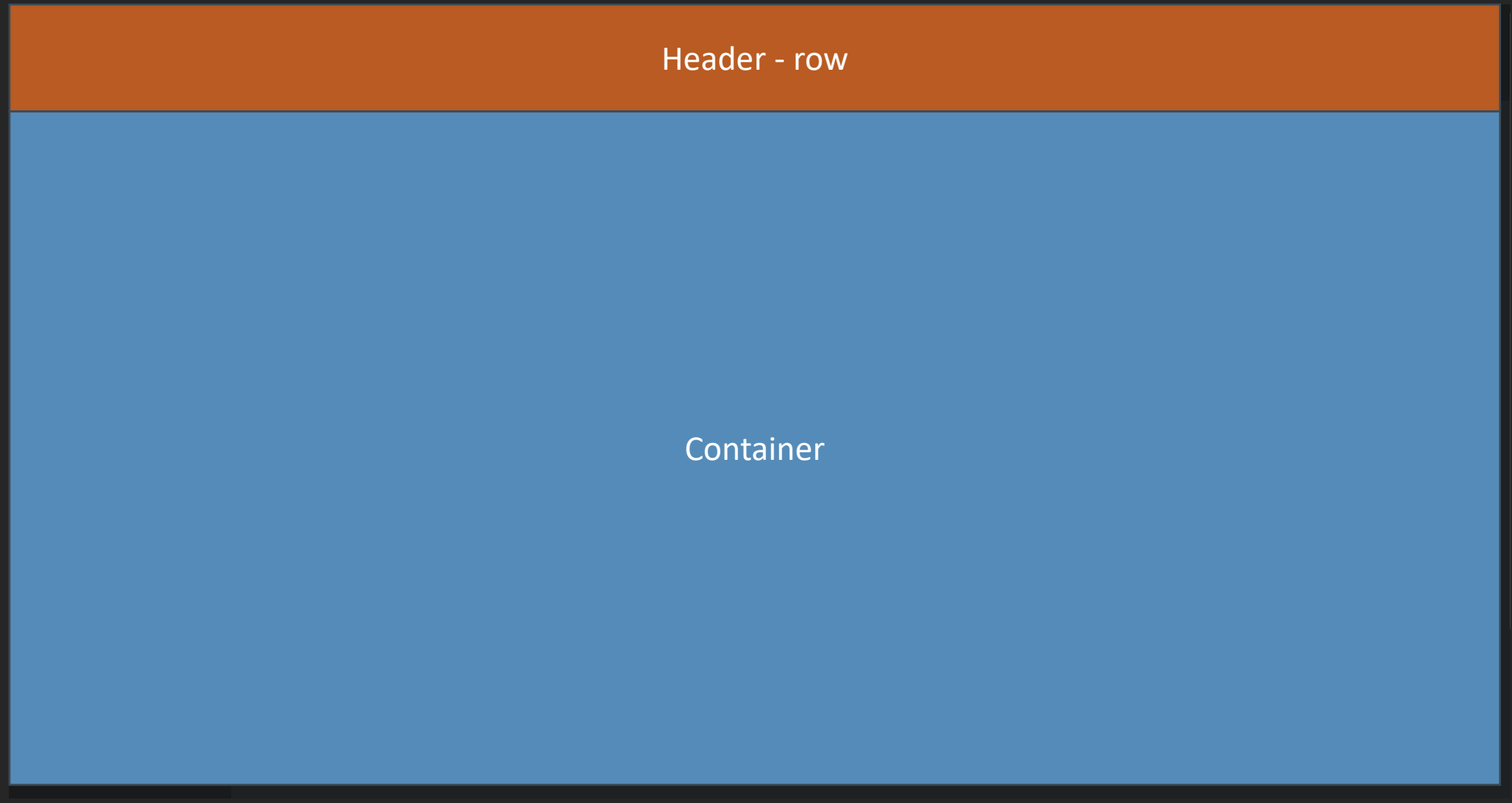
flex: 1;

is a concise way of saying "grow as much as possible, shrink as much as possible, and take up all available space." This is often used to make an element flexible within a flex container, especially when you want it to fill the available space.

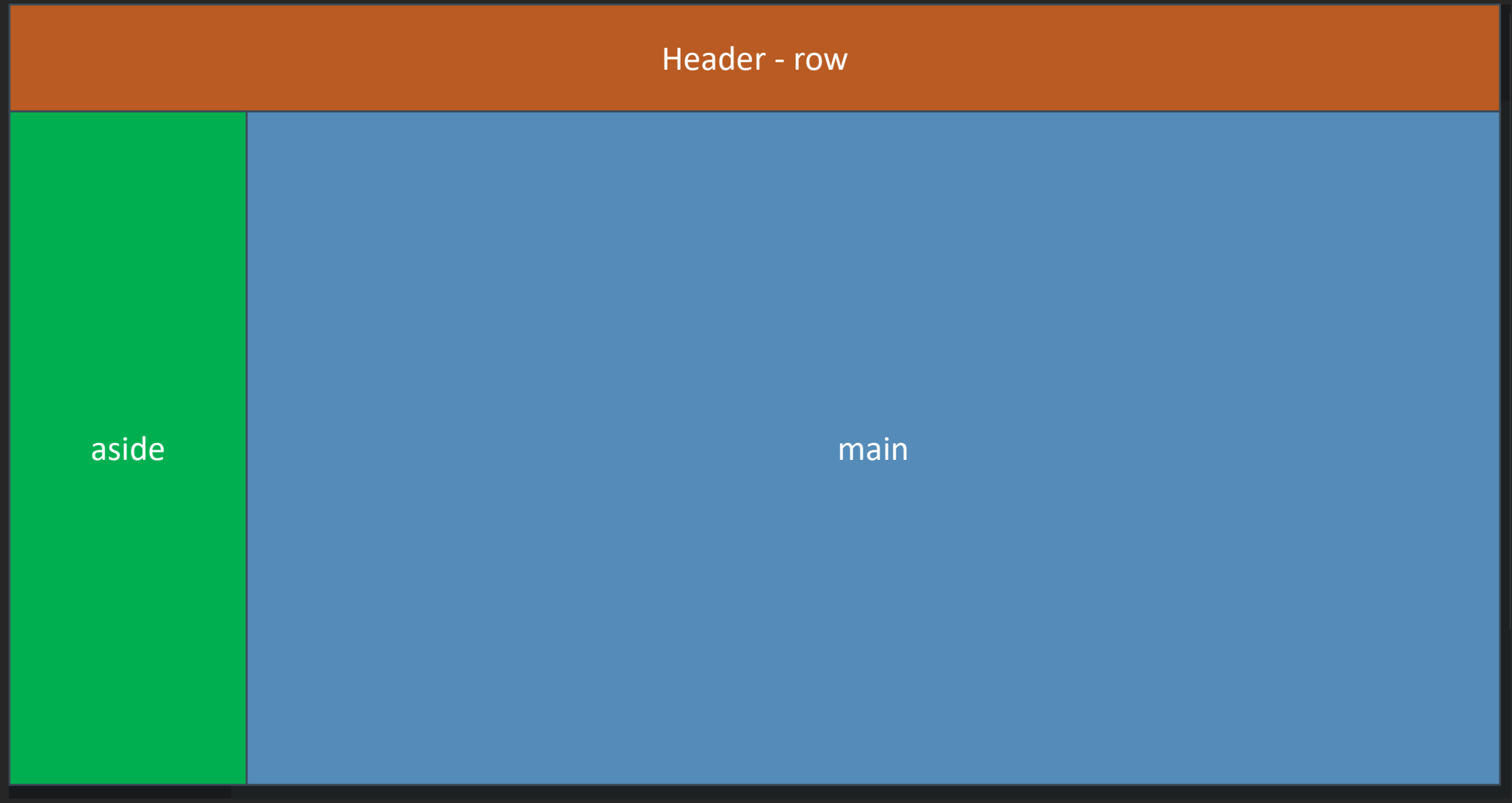
Short hand for:

```
flex-grow: 1;   /* flex item to grow as much as possible */
flex-shrink: 1; /* flex item to shrink as much as possible */
flex-basis: auto; /* Use the default size (auto) as the initial
size (grow with content size)*/
```

Deconstruct a Layout

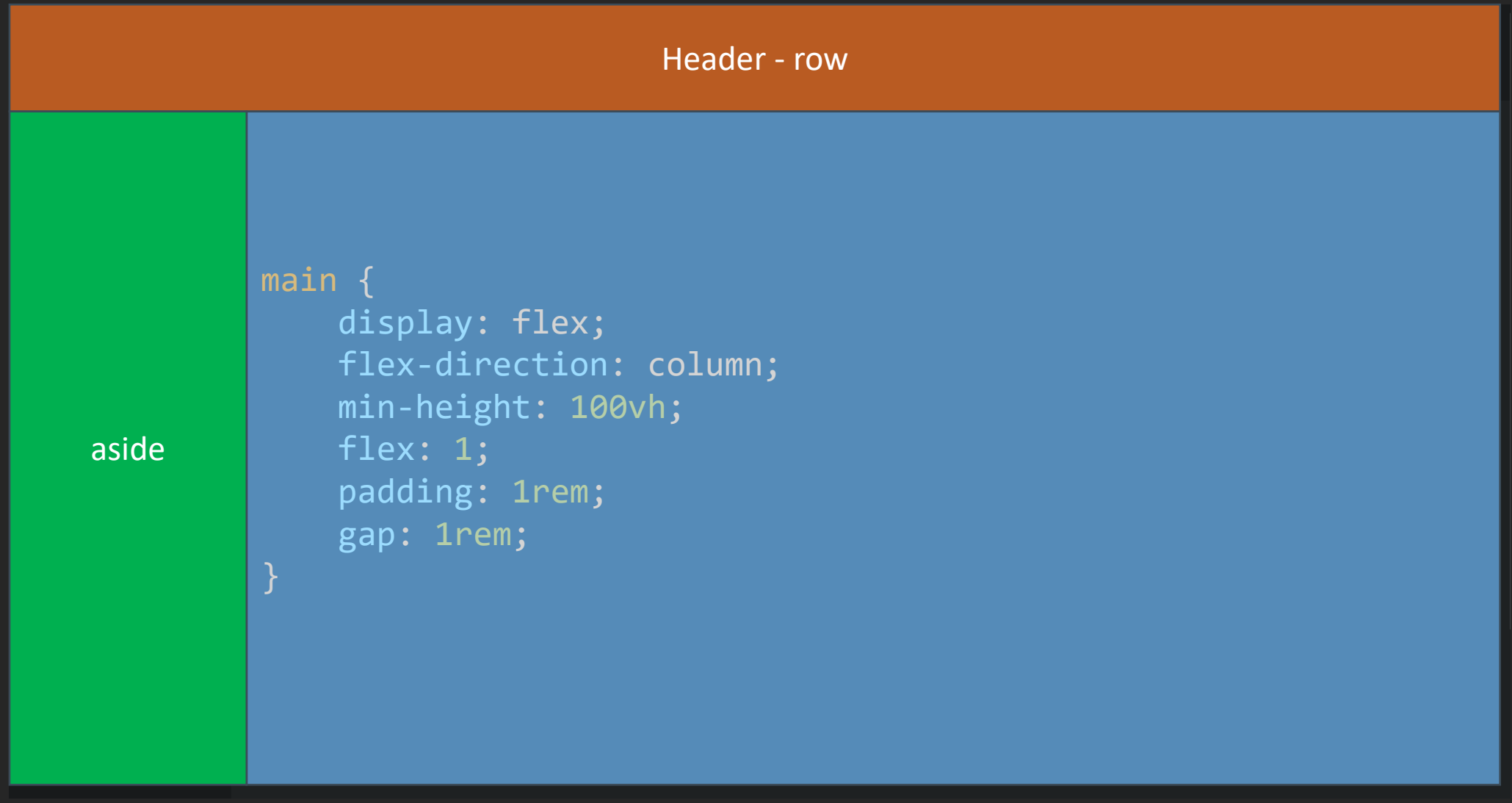


Deconstruct a Layout



```
…
  header{
    background-color: red;
    min-height: 10rem;
  }

  #container{
    background-color: blue;
    display: flex;
    flex: 1;

  }
  aside{
    background-color: green;
    width: 15rem;
    flex: 0 0 14rem;

  }
…
<body>
  <header></header>
  <div id="container">
    <aside>
    </aside>
    <main>
    </main>
  </div>

</body>
```

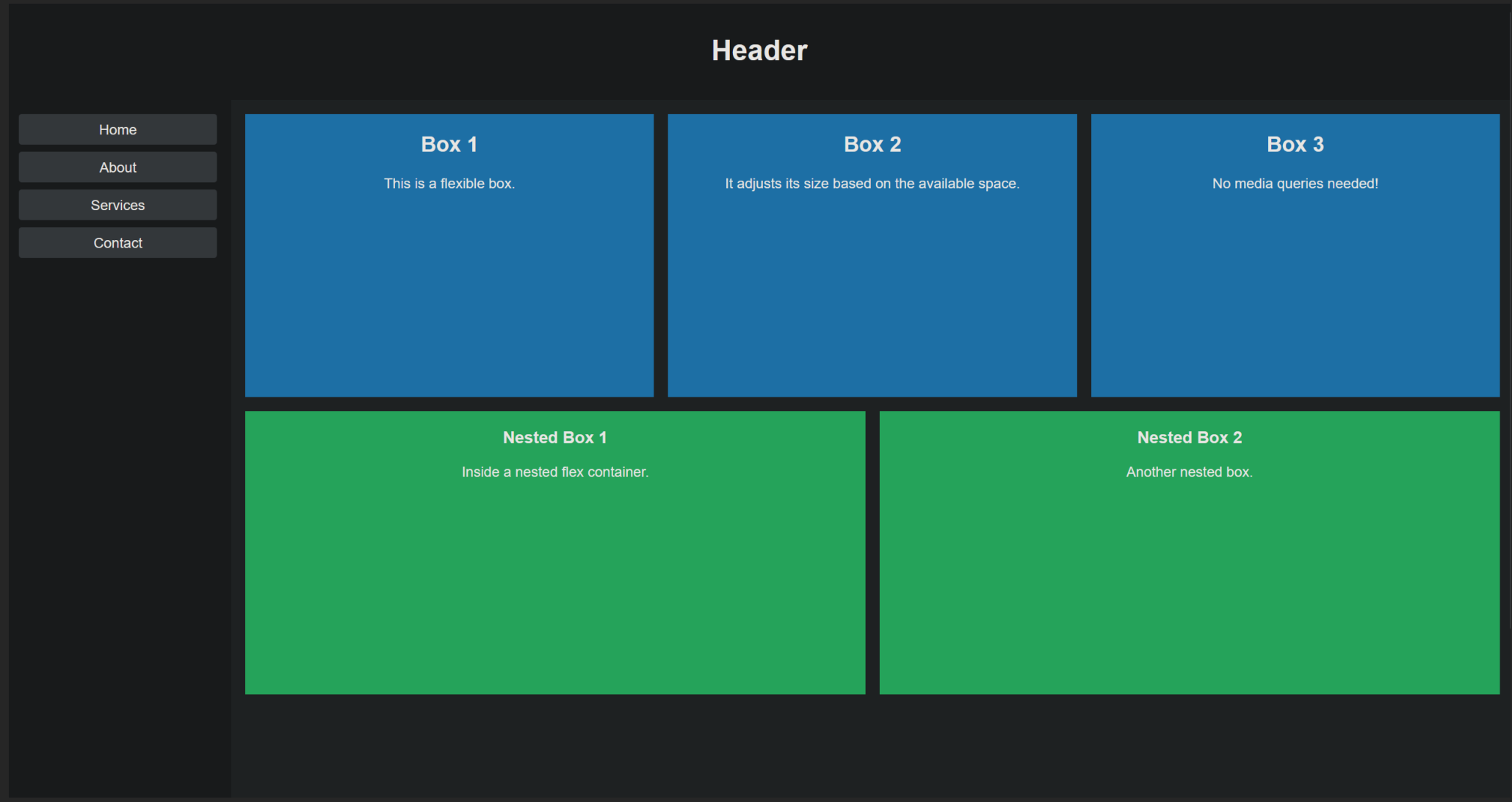
Deconstruct a Layout



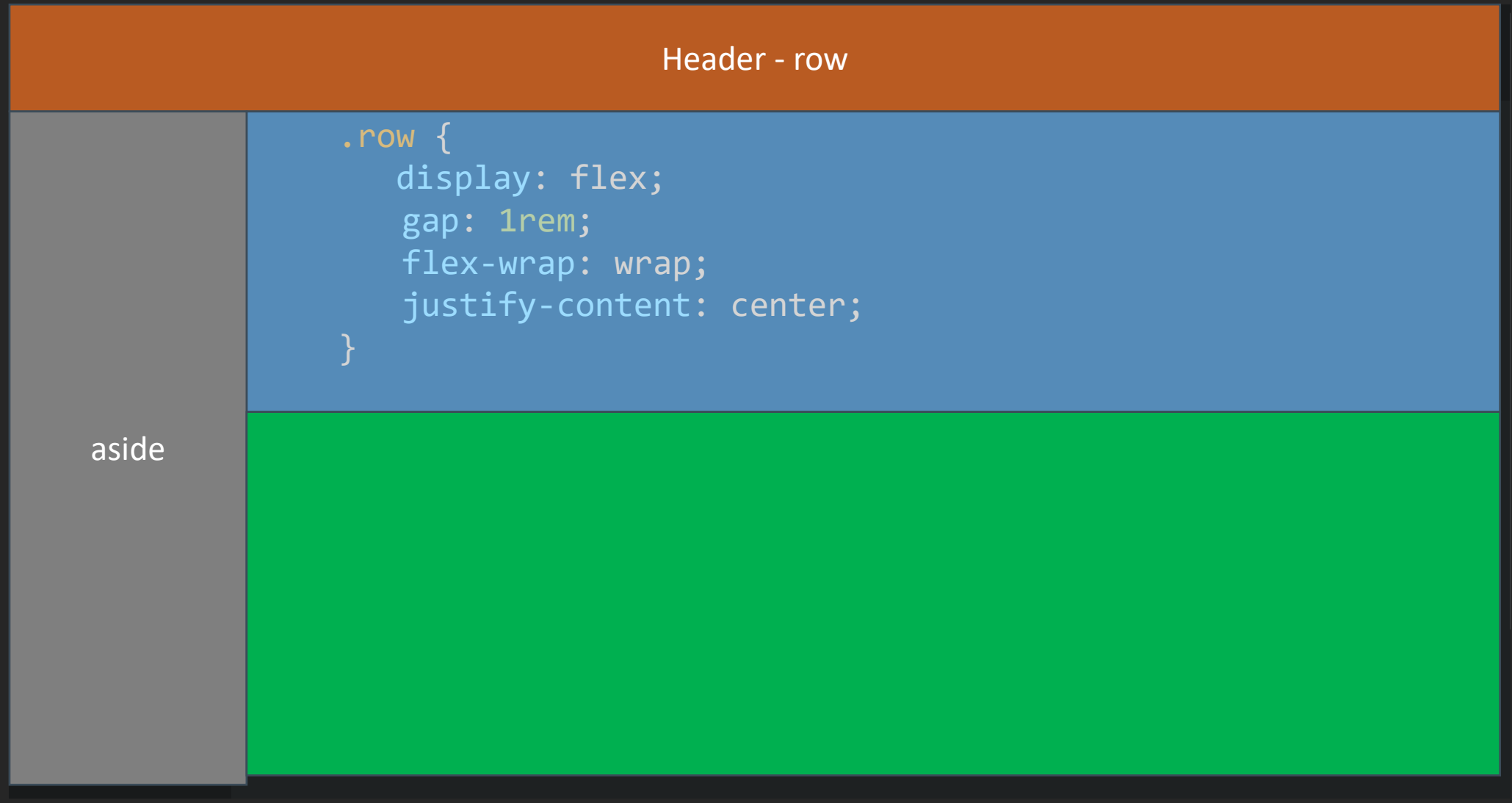
```
…
<body>
  <header></header>
  <div id="container">
    <aside>
    </aside>
    <main>
          <div class="row">  </div>
          <div class="row">  </div>
    </main>
  </div>

</body>
```

Deconstruct a Layout

Deconstruct a Layout



Deconstruct a Layout



```
…
 .row {
        display: flex;
        gap: 1rem;
        flex-wrap: wrap;
        justify-content: center;
   }
…
<body>
  <header></header>
  <div id="container">
    <aside>
    </aside>
    <main>
          <div class="row">  </div>
          <div class="row">  </div>
    </main>
  </div>

</body>
```

```
…
<body>
  <header></header>
  <div id="container">
    <aside>
    </aside>
    <main>
      <div class="row">
        <div class="box"></div>
        <div class="box"></div>
        <div class="box"></div>
      </div>
      <div class="row">
        <div class="box"></div>
        <div class="box"></div>
        <div class="box"></div>
      </div>
    </main>
  </div>

</body>
```

```
.box {
      flex: 1 1 clamp(22rem, 30%, 34rem);
      min-height: 20rem;
      transition: flex-basis 500ms ease-in-out;
      background-color: rgb(71, 71, 71);
      text-align: center;
    }
```

```
<body>
  <header></header>
  <div id="container">
    <aside>
    </aside>
    <main>
      <div class="row">
        <div class="box"></div>
        <div class="box"></div>
        <div class="box"></div>
      </div>
      <div class="row">
        <div class="box"></div>
        <div class="box"></div>
        <div class="box"></div>
      </div>
    </main>
  </div>

</body>
```

The CSS line
`flex: 1 1 clamp(22rem, 30%, 34rem);`
sets the following properties for a flex container or

Flex Grow: The item can grow to fill available space.
Flex Shrink: The item can shrink to fit available space.
Flex Basis: Initial size is set using `clamp(22rem, 30%, 34rem)`, allowing flexibility within the range of 22rem to 34rem, with a preferred size of 30% of the container's size.

Deconstruct a Layout