Three ways to apply CSS to html

1. External
2. Internal
3. Inline

**External CSS**

Html file

<!DOCTYPE html>

<html lang="en">

<head>

 <meta charset="UTF-8">

 <meta http-equiv="X-UA-Compatible" content="IE=edge">

 <meta name="viewport" content="width=device-width, initial-scale=1.0">

 <link rel="stylesheet" href="./styles.css">

 <title>Document</title>

</head>

<body>

 <p>this is  a paragraph</p>

</body>

</html>

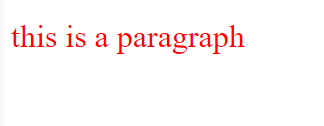
CSS file

p{

 color: red

}

Output



Internal CSS

Inside the head element

 <style>

  p{

   color: limegreen;

  }

If we keep these two in head section

<style>

     p{

      color: green;

     }

    </style>

    <link rel="stylesheet" href="./styles.css">

The last rule will be applied for the p element in html document

Inline CSS

<p style="color: blue;">this is a paragraph</p>

This takes precedence over internal and external CSS

If we make a mistake in CSS the relevant rule will be ignored.

Free online CSS validator: <https://jigsaw.w3.org/css-validator/>

Lesson 2 CSS Selectors

Element selector

body{

 font-size: 22px;

}

p{

 color: purple;

}

Class selector

Html document

<p  class="amitha">Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores, accusantium reprehenderit minus at laborum quam ipsa eum quas sapiente. Incidunt hic, aliquam tenetur provident officiis iste dignissimos quaerat, magnam soluta vero nobis sit. </p>

CSS file

.amitha{

 color: grey;

}

Most commonly used selector, can be used more than once (not like ids)

ID selector

HTML file

<p class="amitha" id="second">Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores, accusantium reprehenderit minus at laborum quam ipsa eum quas sapiente. Incidunt hic, aliquam tenetur provident officiis iste dignissimos quaerat, magnam soluta vero nobis sit. </p>

CSS file

#second{

 font-style: italic;

}

IDs should be unique. Try not to use them much.

Group selection

h1 , h2{

 color: blue;

}

Comma is a must

If it is h1 h2 then the styling will be applied only to h2s nested under h1

Nested selector

HTML file

 <h2>Article 2</h2>

 <p class="amitha" id="second">Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores, accusantium reprehenderit <span>minus</span> dsdfsd </p>

CSS file

p span{

 text-transform: uppercase;

 background-color: gold;

}

It is better to use classes than using nested selectors

HTML file

<p class="amitha">

at laborum <span class="highlight">quam</span> ipsa

    </p>

CSS file

.highlight{

  text-transform: uppercase;

   background-color: green;

}

Universal Selector

Selects everything in html, used to CSS reset

\*{

 font-family: monospace ;

}

Cascading means working like a waterfall. Last rule applies

CSS File

p{

 color: purple;

}

p{

 color: brown;

}

Brown color will be applied for the p element since it is read at last. Specificity can override.

Classes have more specificity than elements. IDs are the most specific ones

**Specificity**

Elements < Classes < IDs

Inheritance

Inheriting styles of parent elements by child elements

Anything related to font and typography are inherited. While others are not.

Example

body{

 font-size: 22px;

}

Font size applied to entire body

body {

  font-size: 22px;

  border: 3px solid green;

}



Border applies to the body only not for other elements.

Note

Form element does not inherit typography settings

But if we want them to inherit we need to select related element then font: inherit;

button, input{

 font: inherit;

}

Universal selector is not related to inheritance. It is selecting all the elements.

We can use body element to write DRY code (don’t repeat yourself). Or we can use html element as well.

We can use main element as well as a semantic element.

We can use a nuclear solution in debugging but it is not advised

p {

  color: purple!important;

}

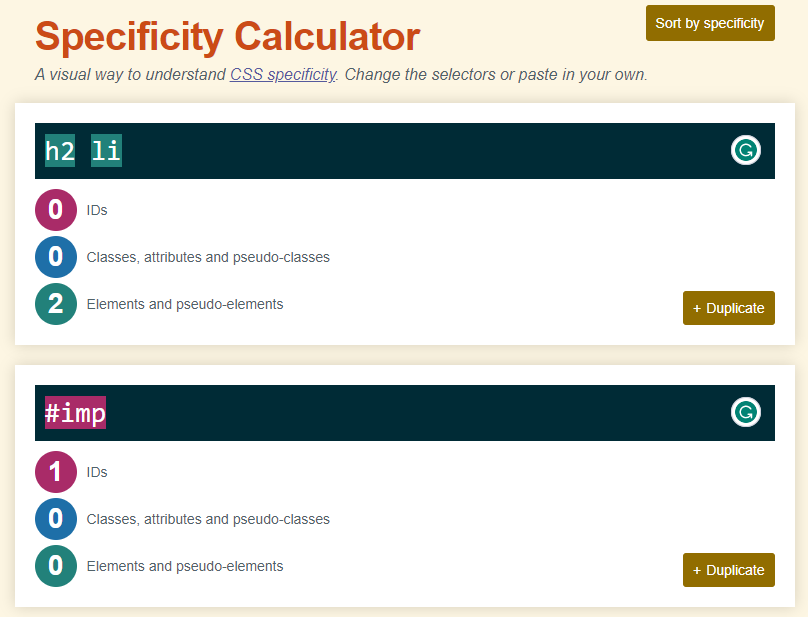
p {

  color: brown;

}

All the text will be converted into purple even font color is changed by classes.

**!important overrides everything. Don’t use it**

Specificity calculator: <https://specificity.keegan.st/>

**Lesson 3 HTML Colors**

In hexadecimal color coding if 2 pairs are matching we can put one letter only

For example

0x000000 = 0x000

0x00ff00 = 0x0f0

0x808080 has no equal pairs

HSL stands for Hue, Saturation and Lightness

Color pallet picking tool: <https://coolors.co/>

Online contrast checker: <https://webaim.org/resources/contrastchecker/> and <https://coolors.co/contrast-checker/>

CSS file

body{

 font-size: 22px;

 font-family: Arial, Helvetica, sans-serif;

 line-height: 1.5;

 background-color: #ffefd5;

 /\* or

 background: green; this is the shorthand method \*/

 color: darkred;

 color: rgb(32, 78, 88);

 /\* a stands for transparency \*/

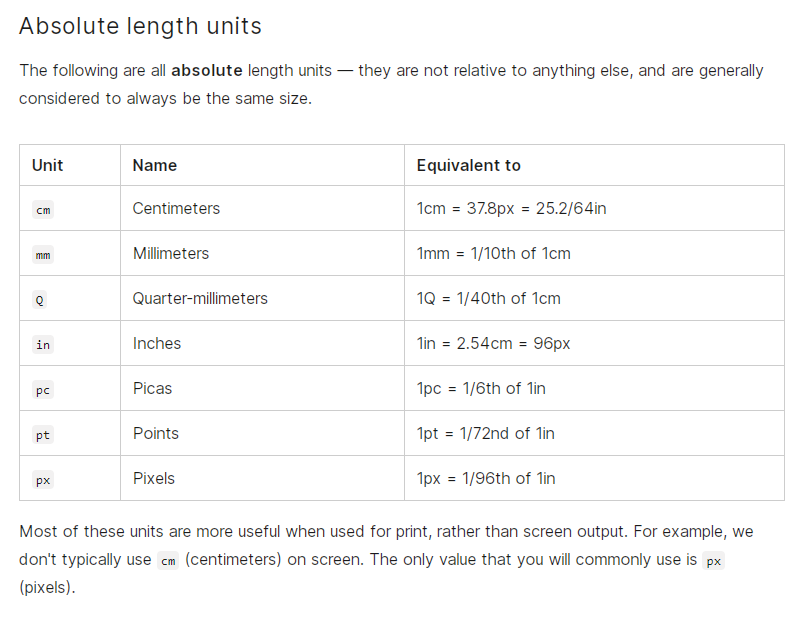
 color: rgba(200, 99, 65, 0.7);

 /\* hexadecimal values \*/

 color: #333;

}

Lesson 4 Units



Default p font size is 16px.

It’s better not to fix font size for p or h elements as it takes users ability to change font size in preferences menu in browser.

Header elements are by default block level elements. They accommodate 100% width of its parent element.

In following example h1 is inside an element called header. Finally the h1 element takes 25% of entire screen

header{

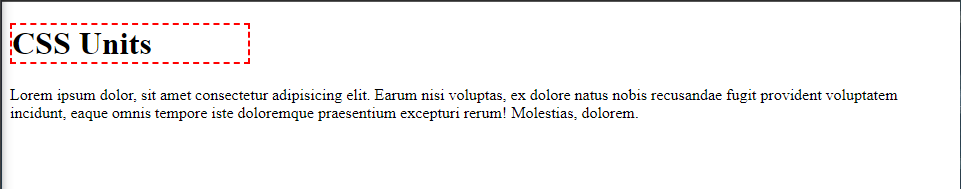
 width: 50%;

}

h1{

 border: 2px dashed red;

 width: 50%;}



Rem – relative to the root element

We should not define font-size : xxpx; in root (html element). We may define it for p element.

p{

 font-size: 1rem;

}

1rem = 16px

2rem = 32px

Rem is only relative to the root element. If there is em it will be four times bigger.

main{

 font-size: 2rem;

}

p{

 font-size: 2em;

}

Resulting font size will be four times bigger.

We may use em units in padding, button padding.

**Ch- character width**

p{

 font-size: 1.5rem;

 width: 40ch;

}

User agent stylesheet is default style settings in google inspect

If we set 100vw for the main element if the content grows in the page we will get a horizontal scroll bar. If we set it to 100% we will not get the horizontal scroll bar if the content grows in the page.

If we want our body element to be 100vh, we should not define it as height: 100vh; but min-height: 100vh; as the content grows it will bring up issues.

**Lesson 5 CSS Box Model**

Starting HTML file

<body>

 <h1>CSS Box Model</h1>

 <main>

  <p>

   Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur blanditiis numquam, libero quasi, sed similique ut expedita praesentium voluptate eum possimus natus quam. Officiis consequatur quas dolorum facere voluptatem exercitationem.

  </p>

 </main>

</body>

CSS file

h1{

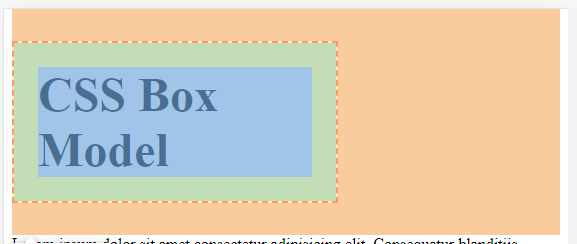
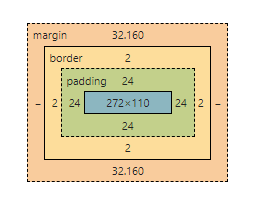
 border: 2px dashed red;

 width: 50%;

 font-size: 3rem;

 padding: 0.5em;

}

CSS Box Model

Em responses for the font size in the relevant element

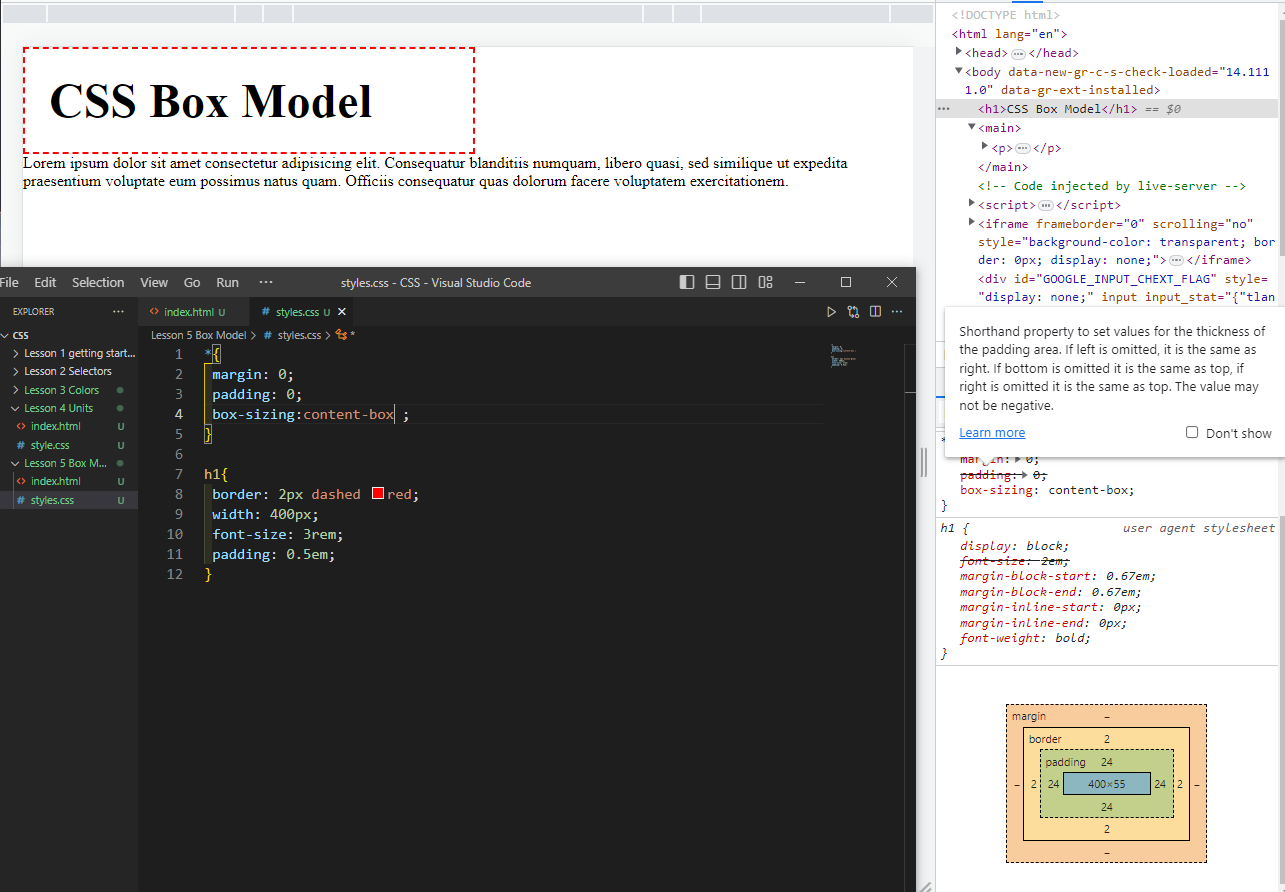
0.67em margin in applied by default. Which means font size affects the default margin value. If we change the h1 font size, the margin and padding px value changes

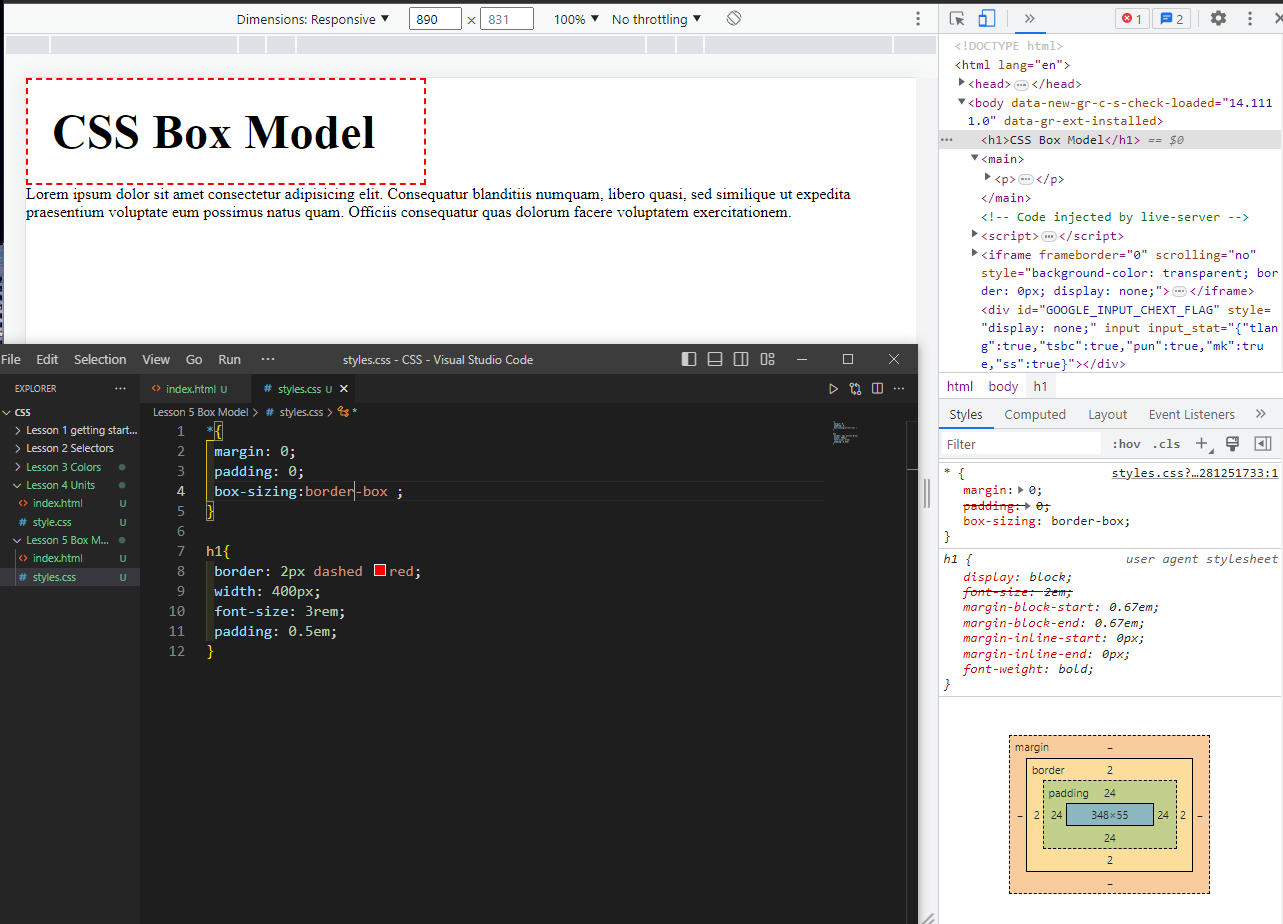
Based on this is good practice to reset margin and padding so that we can take control over them. (CSS reset)

Default box-sizing is content box. Which means if we give a explicit value for width, only the content will get that width. The actual box will be much larger based on margin and padding.

But border-box will arrange content inside by keeping the given width as it is for the external box

Border + Padding + Content = total width

Content-box

border box

Margin and padding

margin-top: 1.5em;

margin-right: 2em;

margin-bottom: 2em;

margin-left: 2em;

/\*margin: 1.5em 2em; top and bottom 1.5em left and right 2em\*/

margin: 1.5em 2em 1.5em 2em; /\*top bottom left right\*/

  /\* margin: 1em;  \*/

  /\* 1em is equal to 1.5rem inside the container since font size is 1.5rem\*/

.container {

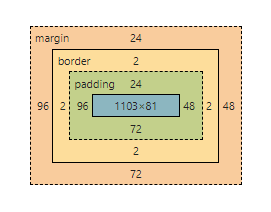
  border: 2px dashed red;

  font-size: 1.5rem;

  margin: 1em 2em 3em 4em;

  padding: 1em 2em 3em 4em;

}



We can separate border properties as well

border-top: 5px solid green;

border-left-width: 10px;

border-left-style: groove ;

border-left-color: blue ;

Outline creates a line outside

outline: 3px solid black;

outline is not calculated border box calculation as it does not take any space

Outline offset can be negative

outline-offset: 5px;

outline-offset: -20px;

draw a circle

HTML file

<div class="circle"></div>

CSS File

.circle{

 background-color: gold;

 margin: 3rem auto; /\*auto keyword is for left and right so it will be centered\*/

 width: 100px;

 height: 100px;

 border: 2px solid red;

 border-radius: 50px; /\*half width and height\*/

 outline: 2px solid black;

 outline-offset: 0.25rem;

}

CSS Typography

Typography is the way that text is arranged and presented

HTML file

<body>

 <header>

  <h1>CSS Typography</h1>

 </header>

 <main>

  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Amet aspernatur beatae reprehenderit officia autem laborum fugiat explicabo aperiam voluptatibus veniam tempora quisquam error, nihil temporibus, velit omnis consequatur! Modi, aliquam!</p>

  <form>

   <label for="name">Name: </label>

   <input type="text" id="name" placeholder="Your Name">

   <button>Submit</button>

  </form>

 </main>

</body>

To inherit font styles to form elements

input, button{

 font: inherit; /\*since input and button are not inherited typography styles\*/

}

Text decoration

Default style is none, used to remove underline from links

p{

 text-decoration: underline 3px solid blue;

}

text-transform: capitalize; /\*capitalize first letter in every word\*/

 text-transform: uppercase; /\*capitalize all the letters\*/

 text-align: justify; /\*uniform in both sides\*/

 text-indent: 2em; /\*indents the first letter to the left\*/

line height default value is 1.2.

1.5 is a good value for better readability

line-height: 1.5;

letter-spacing: 0.1em;

word-spacing: 0.1em;

font-weight: 900;

font-style: italic;

font-family serif, mono-space, sans-serif are common.

 font-family: 'Courier New', Courier, monospace;

Courier and monospace are fallback fonts. If a font has a space in between two words we need to put quotes just like ‘Courier New’.

Web safe fonts are fonts that are installed on most computers and devices. This means that if you use a web safe font, your website will look the same on most computers, regardless of the operating system or browser.

Here are some of the most popular web safe fonts:

* Arial
* Roboto
* Times New Roman
* Courier New
* Verdana
* Tahoma
* Trebuchet MS
* Impact
* Gill Sans
* Georgia
* Palatino
* Baskerville

Getting external fonts from Google

Link section goes to head element, or we can copy starting from @import and put it in the beginning of CSS

Font-family goes to body section

**Lesson 7 Styling Links**

Links are alreay styled. Colored and underlined. When we hover cursor becomes a pointer. Visited links are shown in purple color.

Pseudo classes have more specificity. That’s why even we place a pseudo class on top of an element the pseudo class styling is going to be applied.

Can check with css specificity calculator

The order for link styling

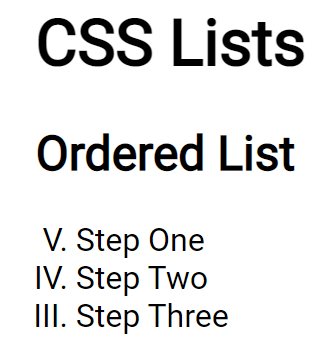
1. “a” element
2. Visited
3. Hover
4. active

focus helps in screen readers.

Hover color can be changed by applying hsl and reducing or increasing h value. Or else we can change opacity

**Lesson 8 Styling Lists**

<ol start="5" reversed>

We must use a numerical numbering to change start value and numbering order. No bullets

HTML body

<body>

    <header>

        <h1>CSS Lists</h1>

    </header>

    <main>

        <article>

            <h2>Ordered List</h2>

            <ol start="5" reversed>

                <li>Step One</li>

                <li>Step Two</li>

                <li>Step Three</li>

            </ol>

        </article>

        <article>

            <h2>Unordered List</h2>

            <ul>

                <li>Step One</li>

                <li>Step Two</li>

                <li>Step Three</li>

            </ul>

        </article>

    </main>

</body>

CSS file

@import url("https://fonts.googleapis.com/css2?family=Roboto&display=swap");

body {

  padding: 5% 10%;

  font-size: 2rem;

  font-family: "Roboto", sans-serif;

}

ol{

  list-style-type: armenian;

  list-style-type: upper-roman;

  list-style-type: none;

  padding: 0;

}

ul{

  list-style-type: square;

  text-align: center;

  list-style-position: inside; /\*some browsers need this line of code otherwise only the text will centered\*/

  color: blue;

  line-height: 1.6;

  list-style-image: url('./checkmark.png'); /\*to add an external image as a bullet\*/

  /\*short hand method and square is the fall back\*/

  list-style: square url('./checkmark.png') inside;

}

/\*to change color of a specific list item\*/

ul li:nth-child(2){

  color: green;

}

/\*to change color of a specific list with odd/even index\*/

ul li:nth-child(odd) {

  color: red;

}

If we want to keep bullet colors black while changing text to some other. First we need to remove bullet styling

::marker{

  color: black;

}

To change the value of a certain list item (ol)

Html  
<article>

            <h2>Ordered List</h2>

            <ol start="5" reversed>

                <li>Step One</li>

                <li value="26">Step Two</li>

                <li>Step Three</li>

            </ol>

CSS

ol{

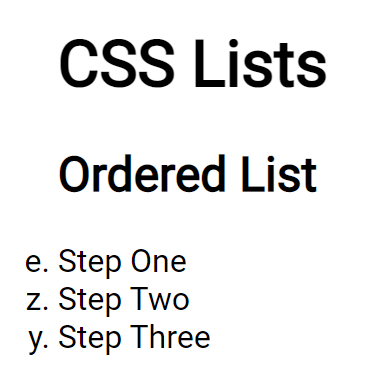
  list-style-type: armenian;

  list-style-type: upper-roman;

  list-style-type: lower-alpha;

  padding: 0;

}



Using ::marker

CSS

ul ::marker{

  color: orange;

  font-family: fantasy;

  font-size: 1em;

  content: "only  $5>>";

}

**CSS Display**

Paragraphs are block level elements. Automatically block level elements consume 100% width stack one on top of another.

Span is an inline element. We cannot apply certain properties to inline elements, such as margin-top, height,

We can apply some properties such as padding, but it may cause to overlap

But if we set display: inline-block everything changes. It respects padding, margin, and height.

Block level elements stack one upon another and they take 100% width (unless inside a specific element with predefined width)

Html body

<body>

    <main>

         <p>This is a paragraph.</p>

        <p>This is <span class="opposite">another</span> paragraph.</p>

    </main>

</body>

CSS file

p{

 background-color: lightgrey;

}

.opposite{

 display: inline-block;

 background-color: #333;

 color: whitesmoke;

 padding: 4rem;

}

To disappear all list items

li{

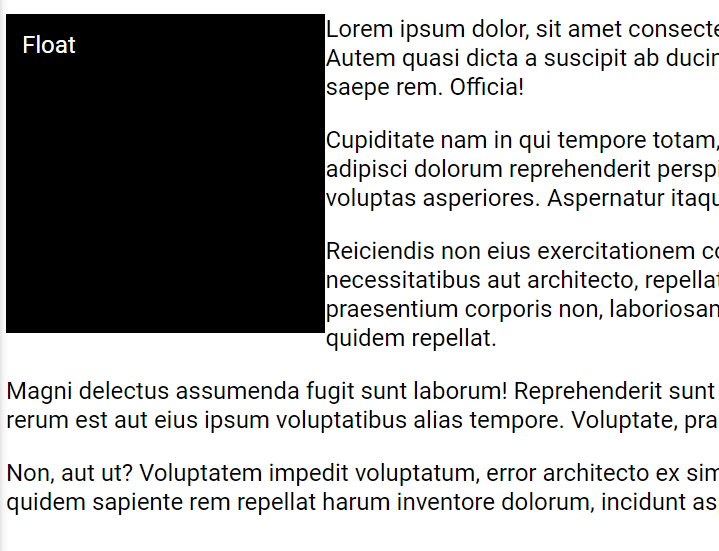
 display: none;

}

CSS Floats

To make 5 p elelments with lorem

P\*5>lorem20

Float is used to wrap around text in an image or square

There we need to more space between float element and text

Applying margin-left for the p element does not give a good output. Instead we must apply margin-right to the class of float element

<body>

    <div class="block left ">Float</div>

    <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Autem quasi dicta a suscipit ab ducimus distinctio ipsum saepe rem. Officia!</p>

    <p>Cupiditate nam in qui tempore totam, iste voluptas. Soluta adipisci dolorum reprehenderit perspiciatis expedita, voluptas asperiores. Aspernatur itaque fugiat veniam!</p>

    <p>Reiciendis non eius exercitationem consectetur necessitatibus aut architecto, repellat incidunt hic! Vel praesentium corporis non, laboriosam ducimus officia quidem repellat.</p>

    <div class="block right ">Float</div>

    <p>Magni delectus assumenda fugit sunt laborum! Reprehenderit sunt expedita voluptas rerum est aut eius ipsum voluptatibus alias tempore. Voluptate, praesentium!</p>

    <p>Non, aut ut? Voluptatem impedit voluptatum, error architecto ex similique laboriosam quidem sapiente rem repellat harum inventore dolorum, incidunt aspernatur.</p>

</body>

CSS

@import url("https://fonts.googleapis.com/css2?family=Roboto&display=swap");

body {

  font-size: 1.5rem;

  font-family: "Roboto", sans-serif;

}

.block{

  width: 30vw;

  height: 30vw;

  background-color: black;

  color: white;

  padding: 1rem;

}

.left{

  float: left;

  margin-right: 1rem;

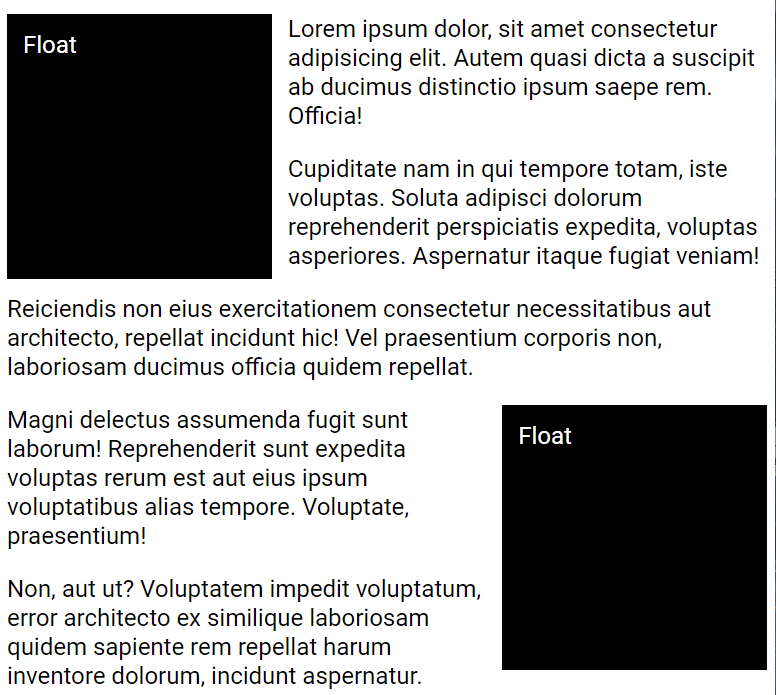
}

.right {

  float: right;

  margin-left: 1rem;

}



To apply text wrapping around float object only to first paragraph

Add empty div element with a class in which clear: both

Html

<div class="block left ">Float</div>

    <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Autem quasi dicta a suscipit ab ducimus distinctio ipsum saepe rem. Officia!</p>

    <div class="clear"></div>

    <p>Cupiditate nam in qui tempore totam, iste voluptas. Soluta adipisci dolorum reprehenderit perspiciatis expedita, voluptas asperiores. Aspernatur itaque fugiat veniam!</p>

CSS

.clear{

  clear: both;

}

If the first paragraph and element are in a specific container if the text inside the container is not enough the container will shrink to accommodate text. It will block the flow of the document

Legacy way is to create a class for the container (section) and change properties to overflow: auto;

New method is change property to display: flow-root;

To make sure container does not shrink based on text content but will accommodate float element add display: flow-root;

**CSS Columns**

When we apply columns for a paragraph we get margin on top and bottom. Specially above first paragraph.

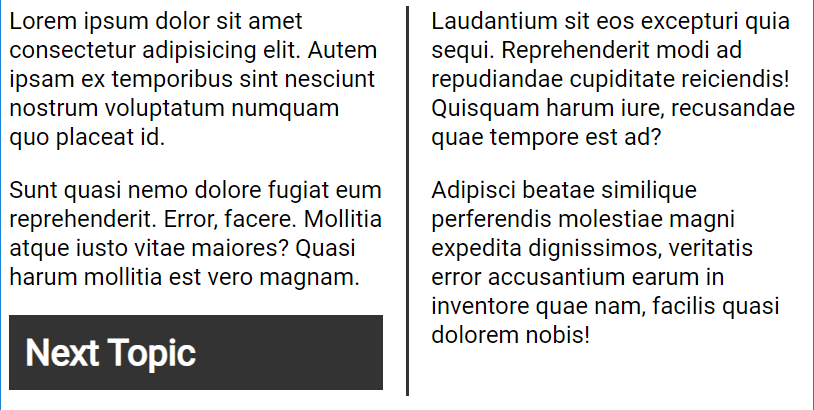
Solution

.columns p{

 margin-top: 0;

}

When we add a h element and add background color and if we shrink content will go into two columns (background). And topic will come to the 1st column while the content is in 2nd column



Solution

.columns h2{

 margin-top: 0;

 background-color: #333;

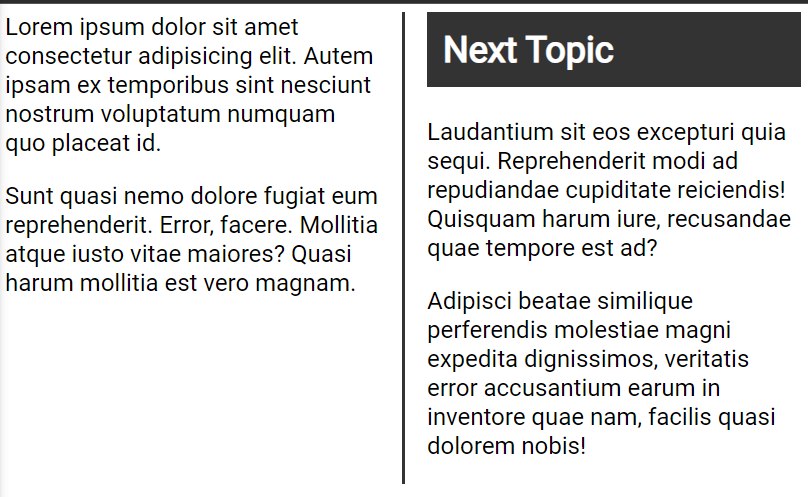
 color: #fff;

 padding: 1rem;

 break-inside: avoid;

 break-before: column;

}



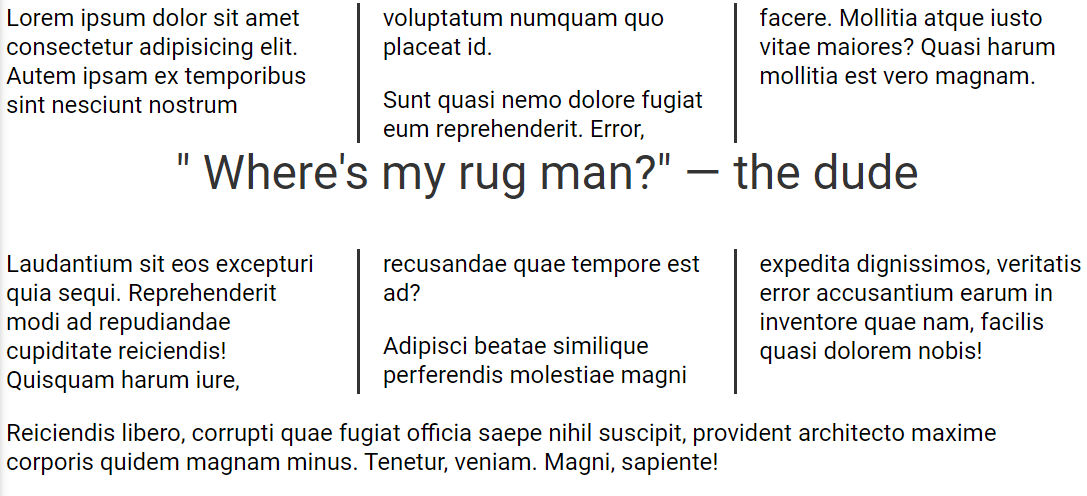
This raise another issue

Not advised to use break-before: column; command

To find symbols

<https://symbl.cc/en/collections/quotation-marks/>

to span a text along columns



Add a certain text in html document under a class(quote)

.quote{

 font-size: 3rem;

 text-align: center;

 color: #333;

 column-span: all;

}

If we apply margin-top to quote class it will not make any change. As we have set column p{margin-top: 0;}. It has more specificity

Solution

.columns .quote{

 font-size: 3rem;

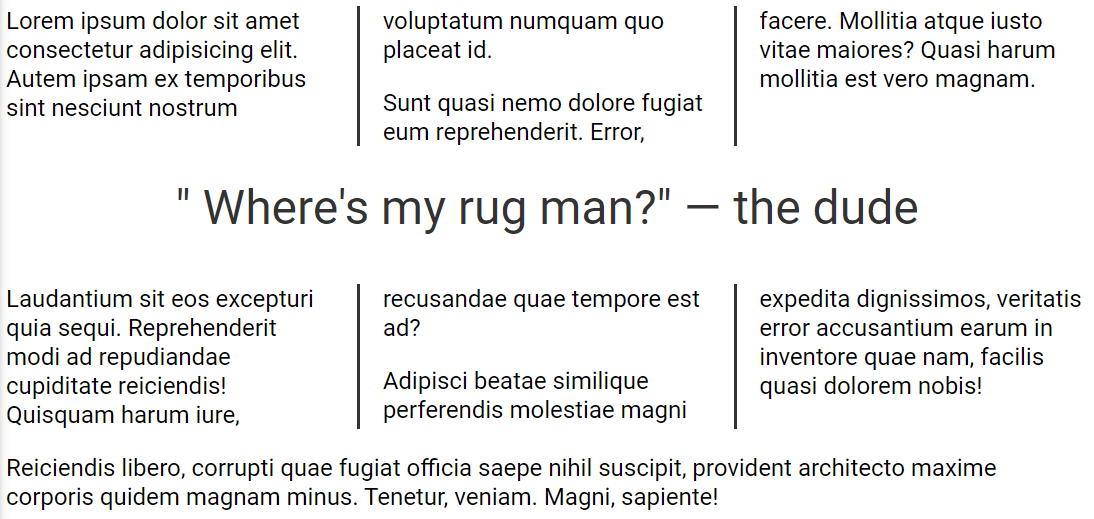
 text-align: center;

 color: #333;

 column-span: all;

 margin-top: 2rem;

}



When we shrink the author line breaks in to second line, to avoid wrapping put required text inside a span element with class (nowrap). In CSS file modify nowrap class

.nowrap{

 white-space: nowrap;

}

**Positions**

Divs are block elements. But not semantic, does not assist assistive readers

 position: static; /\*this is default\*/

.absolute{

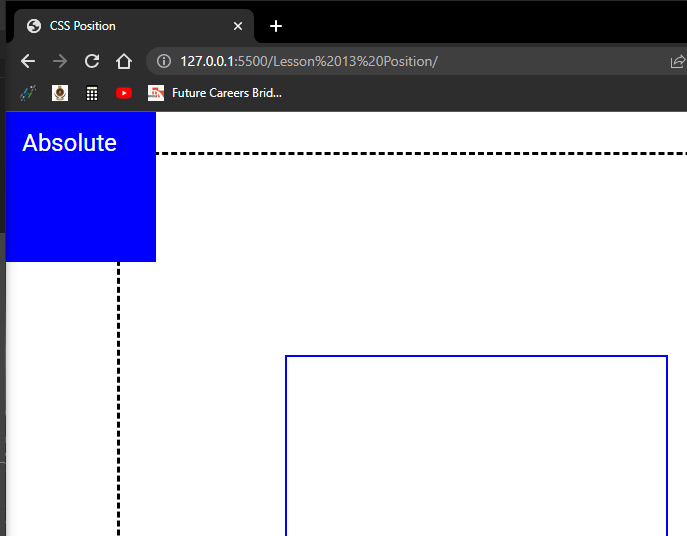
 background-color: #00f;

 position: absolute; /\*this is default\*/

 top: 0px;

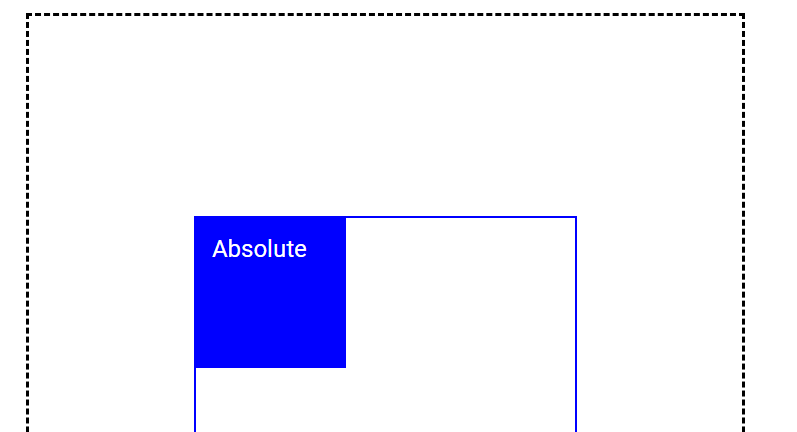
 left: 0px;

}

This will move the container top left of the page irrespectively of the container position

If we do not set top and bottom boundaries to absolute positioning it will take parents position. If we define positions it will take from initial boundary.

Without setting top and bottom



If we add position: relative to a parent container the absolute container will start from that container.

HTML

<body>

    <div class="outer-container">

      <div class="inner-container">

        <div class="box absolute">

          <p>Absolute</p>

        </div>

        <div class="box relative">

          <p>Relative</p>

        </div>

        <div class="box fixed">

          <p>Fixed</p>

        </div>

        <div class="box sticky">

          <p>Sticky</p>

        </div>

      </div>

    </div>

  </body>

CSS

\* {

 margin: 0;

 padding: 0;

 box-sizing: border-box;

}

body {

 font-family: "Roboto", sans-serif;

 font-size: 1.5rem;

 min-height: 200vh;

}

.outer-container {

 border: 3px dashed #000;

 width: 75vw;

 height: 85vh;

 margin: 40px auto;

 position: relative; /\*In here\*/

}

.inner-container {

 border: 2px solid #00f;

 width: 40vw;

 height: 50vh;

 margin: 200px auto;

}

.box {

 width: 150px;

 height: 150px;

 color: #fff;

 padding: 1rem;

}

.absolute{

 background-color: #00f;

 position: absolute;

 top: 0px;

 left: 0px;

}

If we put position: relative in more than one place, the box will go to the closest parent

CSS

.outer-container {

 border: 3px dashed #000;

 width: 75vw;

 height: 85vh;

 margin: 40px auto;

 position: relative; /\*In here\*/

}

.inner-container {

 border: 2px solid #00f;

 width: 40vw;

 height: 50vh;

 margin: 200px auto;

 position: relative;/\*in here as well\*/

}

.box {

 width: 150px;

 height: 150px;

 color: #fff;

 padding: 1rem;

}

.absolute{

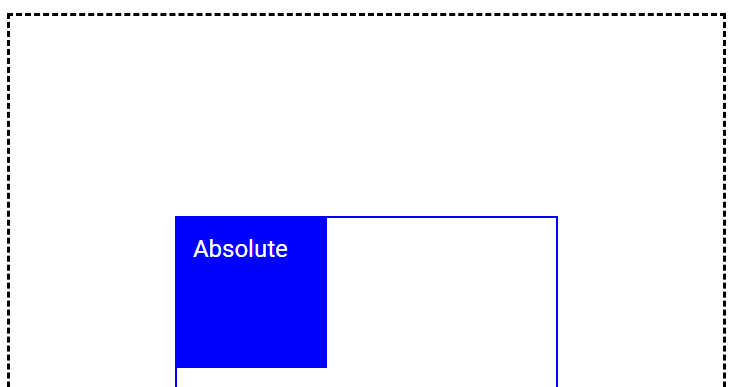
 background-color: #00f;

 position: absolute;

 top: 0px;

 left: 0px;

}



When we set parent position to relative no need to define top and left if we want the box to be in left hand top corner

Relative positioning will be relative to the parent

.relative{

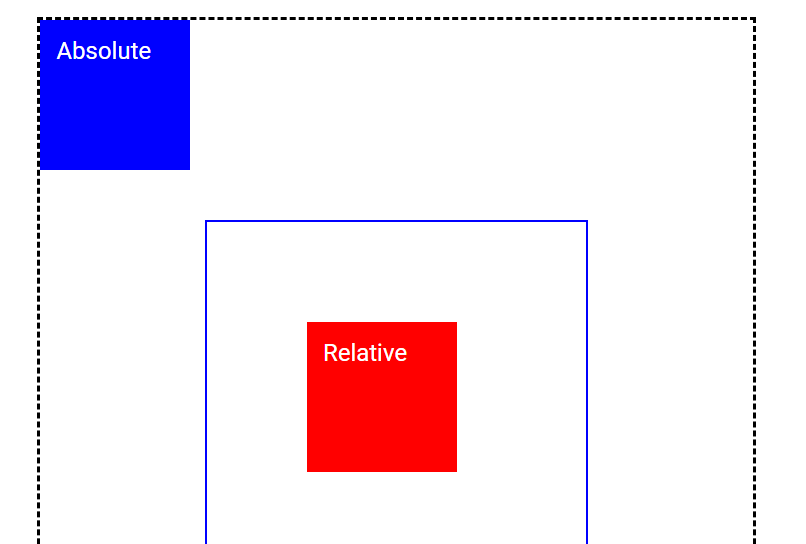
 background-color: #f00;

 position: relative;

 top: 100px;

 left: 100px;

}



.fixed{

 background-color: green;

 position: fixed;

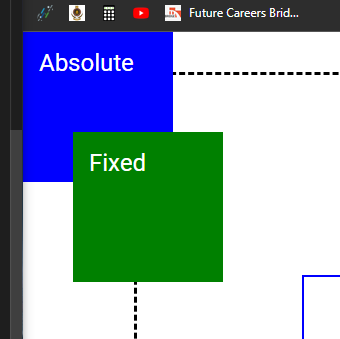
 top: 100px;

}

Fixed one will remain 100px downwards from top even scrolled down

Order in html file will decide what is on top what is on bottom. We can change is by changing z-index.

Before



.absolute{

 background-color: #00f;

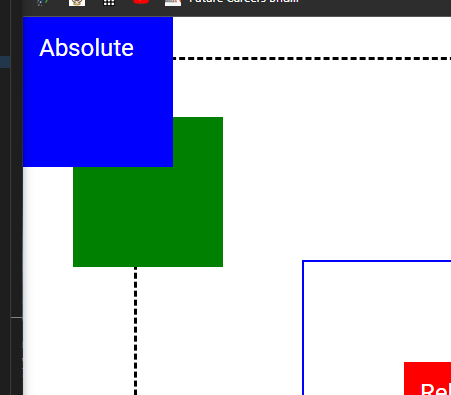
 position: absolute;

 top: 0;

 left: 0;

 z-index: 1;

}

After

Sticky will stay until page is scrolled to the containing container. After that it will go up with the page

.sticky{

 background-color: black;

 position: sticky;

 top: 0;

}

Method to hide things in page

1. display: none
2. opacity: 0
3. Set position to absolute and set left:-1000px. This will visibly remove content from page but still it will be read by screen reader.

.absolute{

 background-color: #00f;

 position: absolute;

 top: 0;

 left: -10000px;

 z-index: 1;

}

First two will not help screen readers

Second part

HTML Body

<button class="social">🚀</button>

    <section id="one">

        <header class="blue">Header One</header>

        <h2>One</h2>

    </section>

    <section id="two">

        <header class="red">Header Two</header>

        <h2>Two</h2>

    </section>

    <section id="three">

        <header class="green">Header Three</header>

        <h2>Three</h2>

    </section>

    <footer>

        <a href="#one">One</a> |

        <a href="#two">Two</a> |

        <a href="#three">Three</a>

    </footer>

CSS

html{

 scroll-behavior: smooth;

}

section{

 height: 100vh;

}

.blue{

 background-color: blue;

}

.red {

 background-color: red;

}

.green {

 background-color: green;

}

header, footer{

 color: white;

 text-align: center;

 height: 100px;

}

header{

 position: sticky;

 top: 0;

 font-size: 5rem;

}

footer{

 background-color: #000;

 position: fixed;

 bottom: 0;

 width: 100%;

 font-size: 3rem;

}

a:visited{

 color: white;

}

.social{

 background-color: royalblue;

 color: #fff;

 font-size: inherit;

 padding: 1rem;

 position: fixed;

 top: 30%;

 left: 0;

 z-index: 1;

}

Flex Box

Div elements are block elements by default.

HTML Body

<body>

   <main class="container">

        <div class="box">1</div>

        <div class="box">2</div>

        <div class="box">3</div>

        <div class="box">4</div>

        <div class="box">5</div>

        <div class="box">6</div>

   </main>

</body>

CSS

@import url("https://fonts.googleapis.com/css2?family=Roboto&display=swap");

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

body {

  font-family: "Roboto", sans-serif;

  min-height: 100vh;

  padding: 20px;

}

.container {

  max-width: 800px;

  min-height: 400px;

  margin-inline: auto;

  border: 1px solid #000;

}

.box {

  min-width: 100px;

  min-height: 100px;

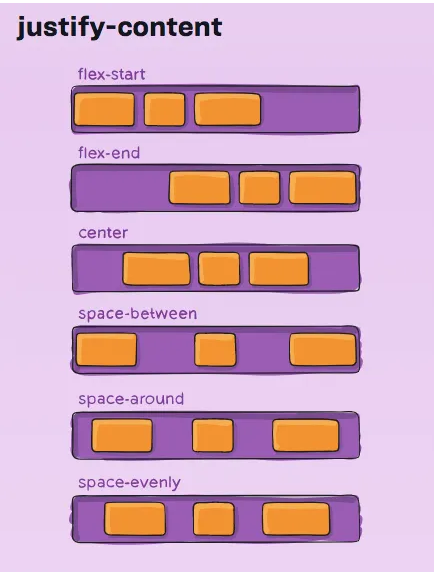
  background-color: #000;

  color: #fff;

  font-size: 2rem;

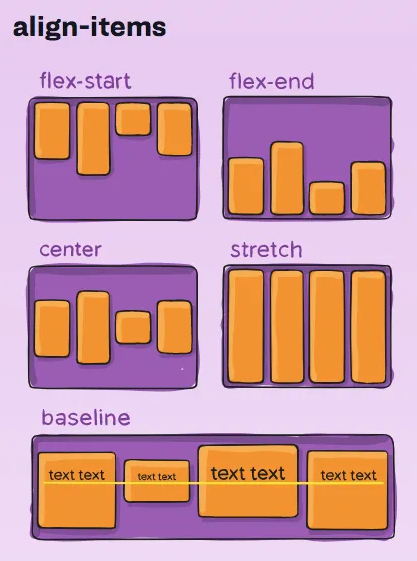
  padding: 0.5rem;

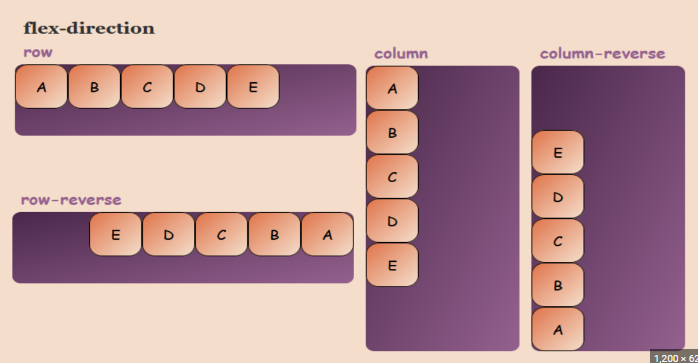
}

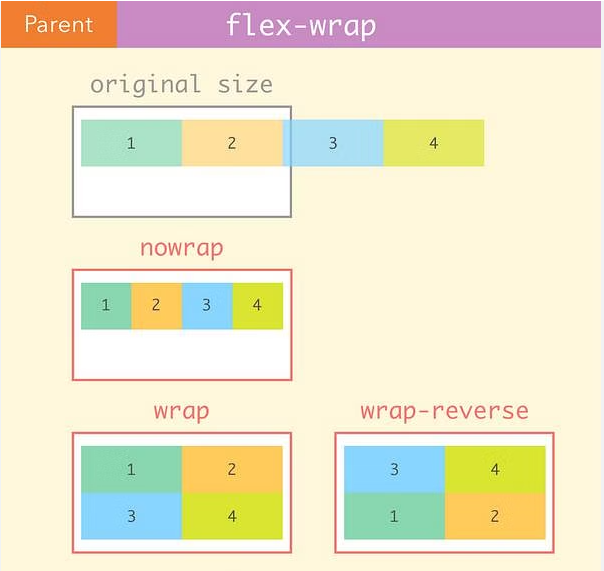
Justify content in flexbox

We can define gap between flex items by

gap: 1rem;

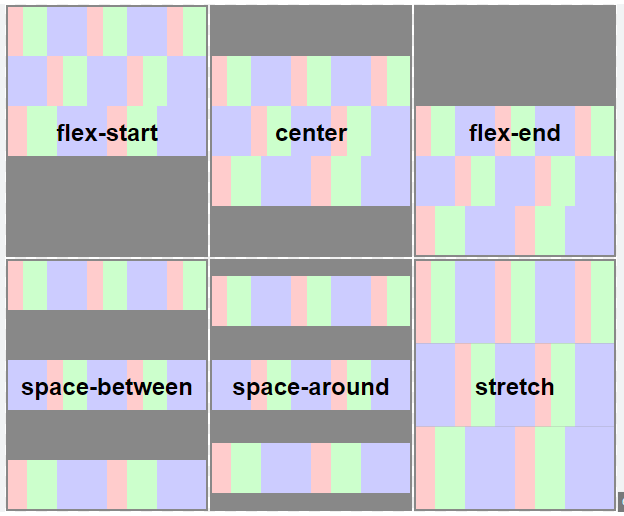
Align items

Flex-direction

Flex-wrap

Short hand for flex direction and flex-wrap

flex-flow: row wrap;

Align-content

We can edit the flex items as well by adding display: flex in to the flex items.

.box {

  min-width: 100px;

  height: 100px;

  background-color: #000;

  color: #fff;

  font-size: 2rem;

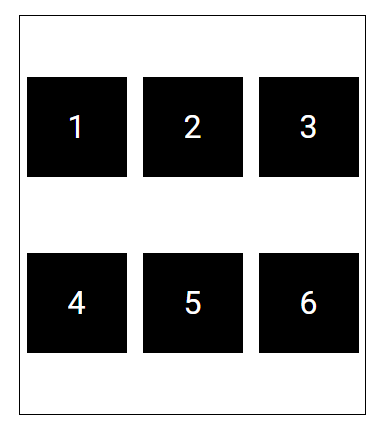
  padding: 0.5rem;

  display: flex;

  justify-content: center;

  align-items: center;

}

Output

Code to center any element inside a container

  display: flex;

  justify-content: center;

  align-items: center;

Comment out last four divs in html and min-width : 100px in css

.box {

  /\* min-width: 100px; \*/

  height: 100px;

  background-color: #000;

  color: #fff;

  font-size: 2rem;

  padding: 0.5rem;

  display: flex;

  justify-content: center;

  align-items: center;

  flex-grow: 1;

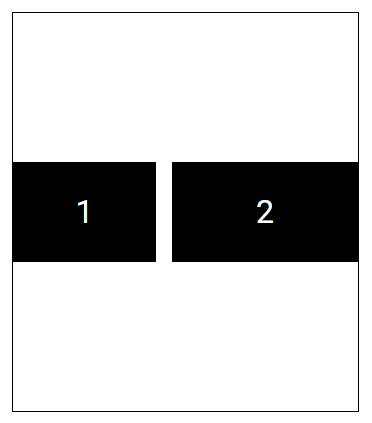
  flex-basis: 100px;

}

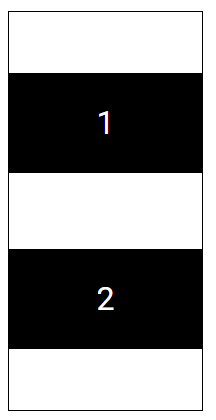
.box:nth-child(2){

  flex-grow: 2;

}

Output

1 and 2 flex items will share empty space in flex container by proportions of 1/3 and 2/3.

When the screen is shrinked

Flex-shrink

.container {

  max-width: 800px;

  min-height: 400px;

  margin-inline: auto;

  border: 1px solid #000;

  display: flex;

  justify-content: center;

  align-items: center;

  flex-flow: row nowrap;

  align-content: space-evenly;

  gap: 1rem;

}

.box {

  /\* min-width: 100px; \*/

  height: 100px;

  background-color: #000;

  color: #fff;

  font-size: 2rem;

  padding: 0.5rem;

  display: flex;

  justify-content: center;

  align-items: center;

  flex-shrink: 1;

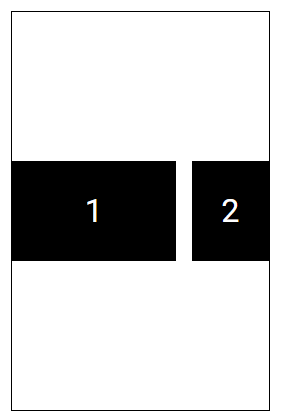
  flex-basis: 250px;

}

.box:nth-child(2){

  flex-shrink: 2;

}

Output

Exact opposite of flex-grow

Flex grow lets something grow beyond flex-basis value.

Flex shrink lets something shrink below flex-basis value.

Short hand method

flex: grow shrink basis;

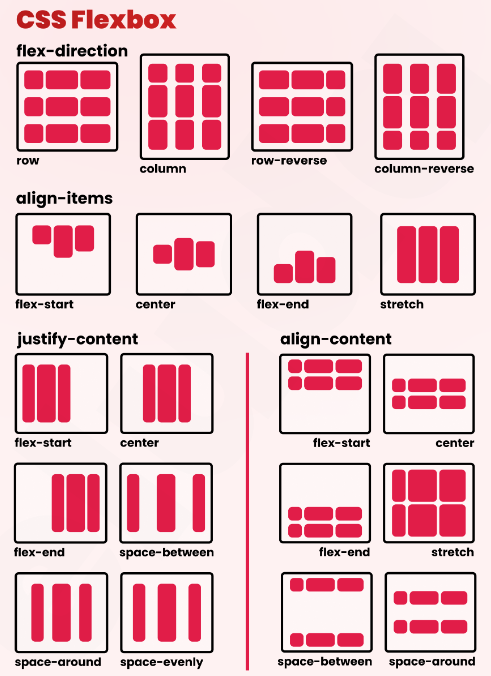
We can use percentages as well

flex: 1 1 40%;

Good place to learn css flex box

<https://flexboxfroggy.com/>

flexbox on one image



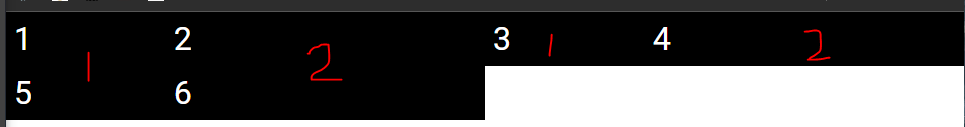
CSS Grid

In grid-template-column: xx ; we can use px values, combination of px and fr, and fr (fraction) values as well.

If column widths are equal we can use

grid-template-columns: repeat(3, 1fr);

  grid-template-columns: repeat(2, 1fr 2fr);



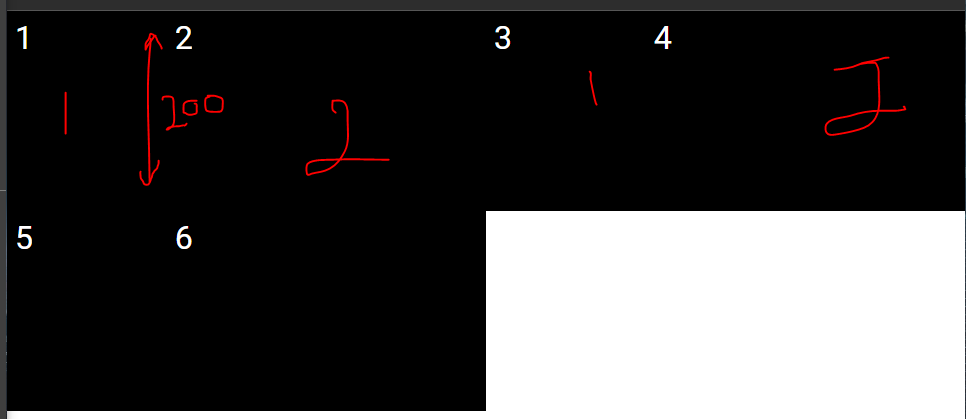
.container{

  min-height: 400px;

  display: grid;

  grid-template-columns: repeat(2, 1fr 2fr);

  grid-auto-rows: minmax(150px, auto);



.container{

  min-height: 400px;

  display: grid;

  grid-template-columns: repeat(2, 1fr 2fr);

  grid-auto-rows: minmax(150px, auto);

  row-gap: 1em;

  column-gap: 1em;

}

Short hand method for gpa

  gap: row\_gap column\_gap;

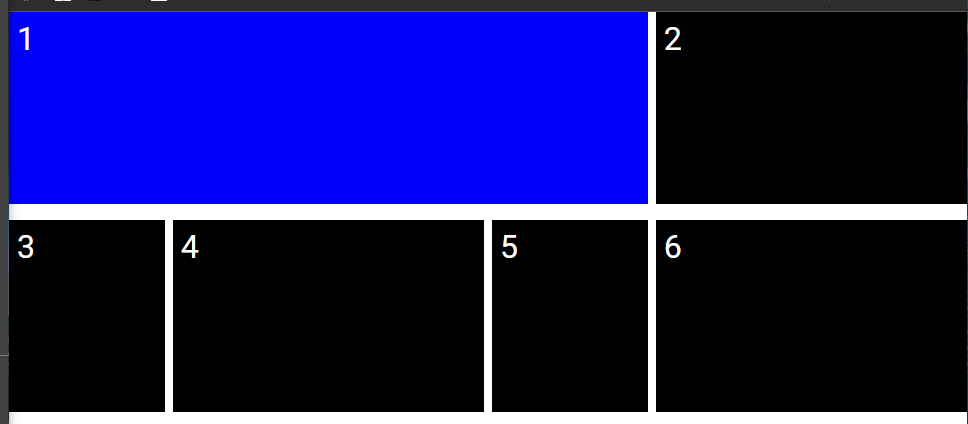
.box:first-child{

  background-color: blue;

  grid-column-start: 1;

  grid-column-end: 4;

}



.box:first-child{

  background-color: blue;

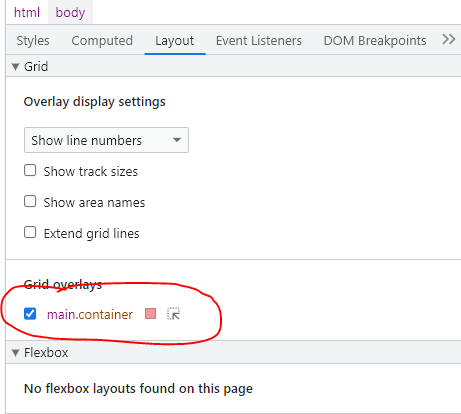
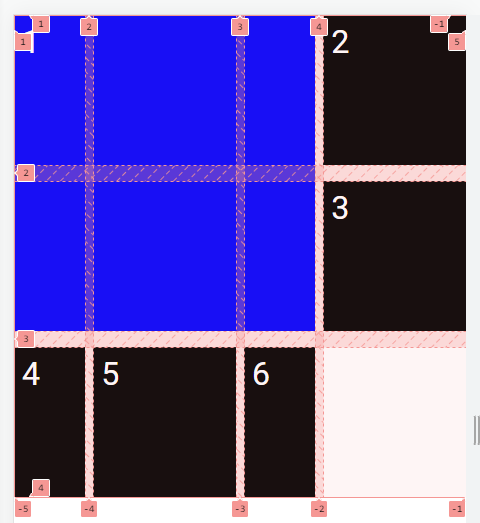
  grid-column-start: 1;

  grid-column-end: 4;

  grid-row-start: 1;

  grid-row-end: 3;

}



Grid using short hand method

.box:first-child{

  background-color: blue;

  grid-column: 1/4;

  grid-row: 1/3;

}

Just like flex box we can make grid inside a grid element and center the content inside.

.box:first-child{

  background-color: blue;

  grid-column: 1/4;

  grid-row: 1/3;

  display: grid;

  justify-content: center;

  align-content: center;

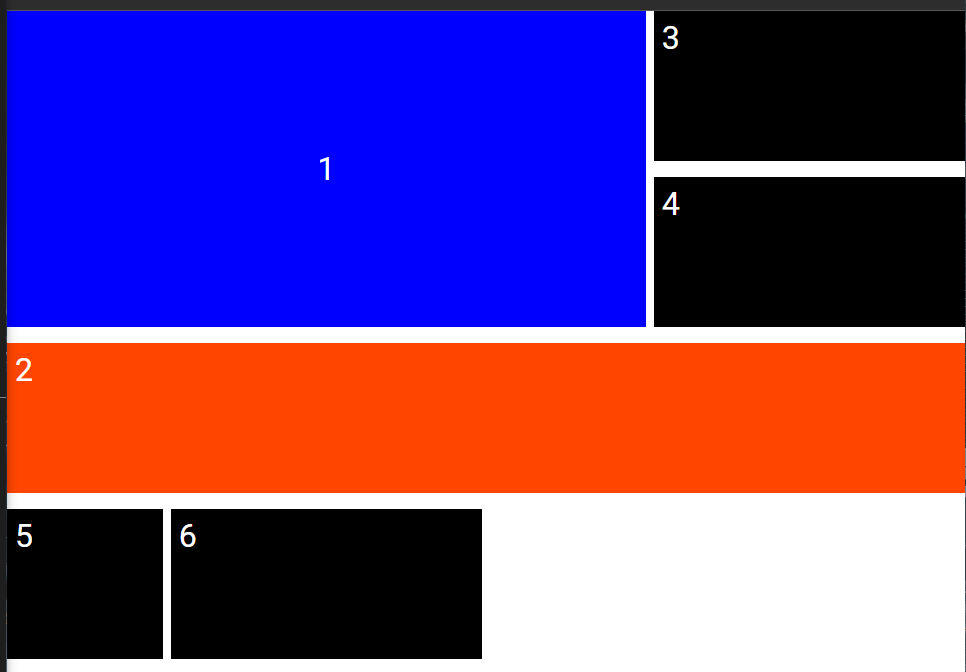
}

.box:nth-child(2) {

  background-color: orangered;

  grid-column: 1/5;

  grid-row: 3/4;



short hand method for

justify-content: center;

align-content: center;

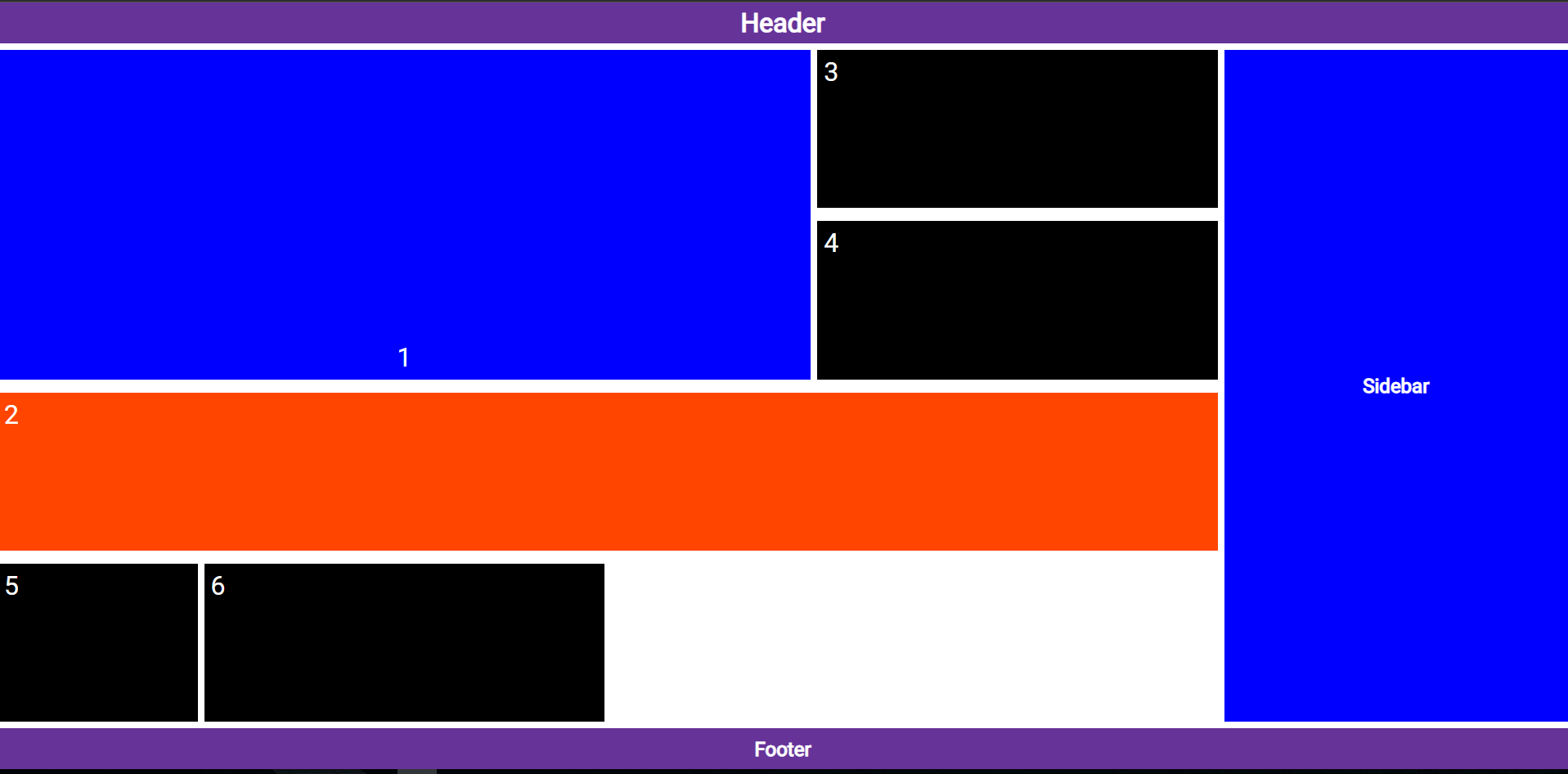
is

place-content: align\_content justify\_content;

we can place center if both properties needs to be center.

  place-content: center;

Output



HTML body

 <header class="header el"><h1>Header</h1></header>

    <main class="container">

            <div class="box">1</div>

            <div class="box">2</div>

            <div class="box">3</div>

            <div class="box">4</div>

            <div class="box">5</div>

            <div class="box">6</div>

    </main>

    <aside class="sidebar el"><h2>Sidebar</h2></aside>

    <footer class="footer el"><h2>Footer</h2></footer>

CSS

@import url("https://fonts.googleapis.com/css2?family=Roboto&display=swap");

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

body {

  font-family: "Roboto", sans-serif;

  min-height: 100vh;

  display: grid;

  grid-template-columns: repeat(9, 1fr);

  grid-auto-rows: 50px auto 50px;

  grid-template-areas:

    "hd hd hd hd hd hd hd hd hd "

    "mn mn mn mn mn mn mn sb sb "

    "ft ft ft ft ft ft ft ft ft ";

    gap: 0.5rem;

}

.el{

  background-color: rebeccapurple;

  color: white;

  display: grid;

  place-content: center;

}

.header{

  grid-area: hd;

}

.sidebar{

  grid-area: sb;

  background-color: blue;

}

.footer{

  grid-area: ft;

}

.container{

  grid-area: mn;

  min-height: 400px;

  display: grid;

  grid-template-columns: repeat(2, 1fr 2fr);

  grid-auto-rows: minmax(150px, auto);

  row-gap: 1em;

  gap: 1em 0.5em

}

.box {

  background-color: #000;

  color: #fff;

  font-size: 2rem;

  padding: 0.5rem;

}

.box:first-child{

  background-color: blue;

  grid-column: 1/4;

  grid-row: 1/3;

  display: grid;

  /\* justify-content: center;

  align-content: center; \*/

  place-content: end center;

}

.box:nth-child(2) {

  background-color: orangered;

  grid-column: 1/5;

  grid-row: 3/4;

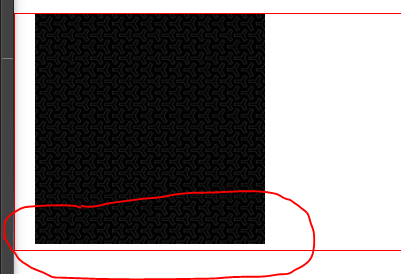
}

Lesson 16 Images

When we add an image it is necessary to define related height and width with no units in html file.

Images are inline elements by default.

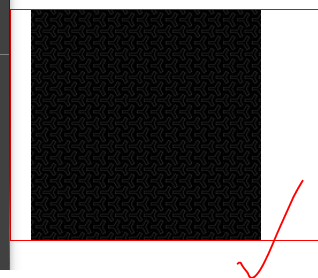
By default a space will be added after an image. It’s because it’s an inline element

To get rid of

img{

  display: block;

}

Output

In css we can define width to any amount even if we have defined original image’s width and height. When we define width and height in css height will have auto

.profile-pic-figure img{

  width: 100%;

  height: auto;

}

Contain will go with image repeat. So its not advised to use it. Cover is good

text-shadow: 2px 2px 5px black;

text shadow is a good thing to use on images to pop out text

x-direction y-direction blur-amount shadow-color

another way is we can add hsl background color

background-color: hsla(0, 0%, 0%, 0.303);

or else we can a background mask

  background-color: hsla(0, 0%, 0%, 0.35);

we can set background image position. Its better to use

background-position: center center;

Output



HTML Body

<div class="container">

      <section class="hero">

        <figure class="profile-pic-figure">

          <img

            src="./img/profile-800x800.png"

            alt="profile picture"

            width="800"

            height="800"

            title="My profile picture"

          />

          <figcaption class="offscreen">Jane Doe</figcaption>

        </figure>

        <h1 class="h1">

          <span class="nowrap">Hello 😀</span>

          <span class="nowrap">I'm Jane</span>

        </h1>

      </section>

    </div>

CSS

@import url("https://fonts.googleapis.com/css2?family=Nunito&display=swap");

/\*begin reset\*/

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

img{

  display: block;

}

/\*end reset\*/

/\*utility classes\*/

.nowrap{

  white-space: nowrap;

}

/\*end utility classes\*/

.offscreen{

  position: absolute;

  left: -10000px;

}

body {

  font-family: "Nunito", sans-serif;

  min-height: 100vh;

}

.container{

  background-color: rgb(251, 210, 156);

    background-image: url(./img/map-2176x1451.png);

    background-repeat: repeat;

    background-size: cover;

    background-position: center center;

}

.hero{

  border-bottom: 2px solid black;

  padding: 20px;

  display: flex;

  justify-content: flex-start;

  align-items: center;

  gap: 30px;

  background-color: hsla(0, 0%, 0%, 0.35);

}

.h1{

  font-size: 500%;

  color: white;

  text-shadow: 2px 2px 5px black;

  /\* background-color: hsla(0, 0%, 0%, 0.303);

  padding: 0.5rem;

  border-radius: 10px; \*/

}

Output

HTML body

Nothing special

CSS

body {

  font-family: "Nunito", sans-serif;

  min-height: 100vh;

  background-color: aliceblue;

  background-image: url("./img/bubbles.png"),

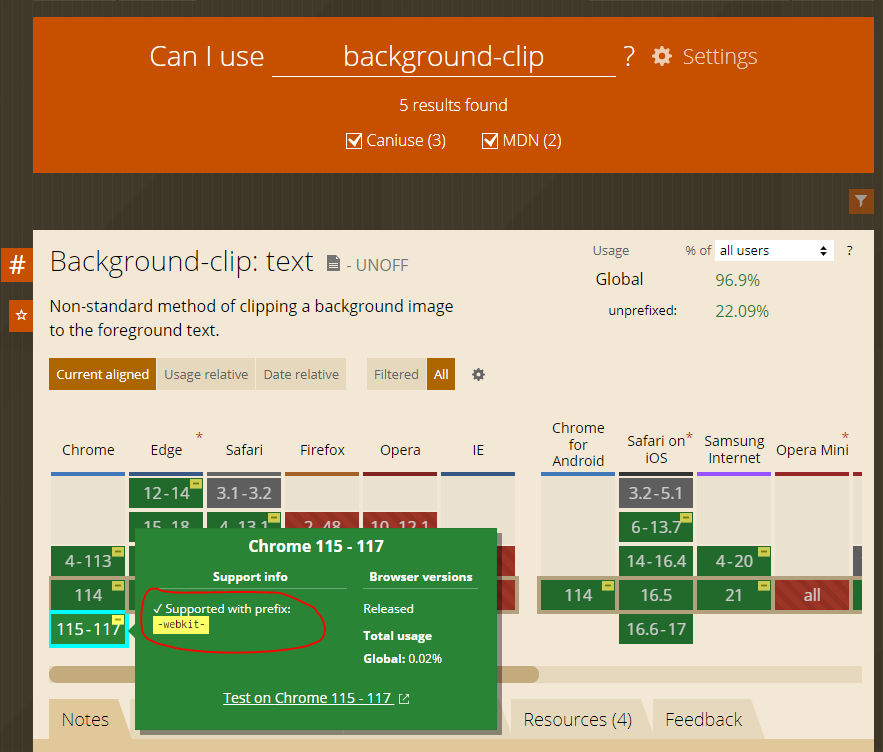
                    linear-gradient(to left, steelblue, white);

  background-repeat: repeat-y, no-repeat;

  background-position: right center;

  background-size: 20%, auto;

}

Support tables for HTML CSS

Link: <https://caniuse.com/>

Output

HTML

 <section>

        <p class="clip">Jane</p>

    </section>

CSS

.clip{

  font-weight: 800;

  font-size: 18rem;

  text-align: center;

  background-image: url("./img/scenic-2200x1331.png");

  background-size: 100%;

  text-transform: uppercase;

  /\*for chrome with prefix\*/

  -webkit-background-clip: text;

  /\*for firefox\*/

  background-clip: text;

  color: hsla(0, 0%, 0%, 0.2);

}

Instead of these lines in previous code we can use short hand method as well

background: repeat-y right center url("./img/bubbles.png"), no-repeat linear-gradient(to left, steelblue, white);

  background-size: 20%, auto;

Lesson 17 Media Queries

This line is necessary to activate responsiveness

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Syntax:

@media media type and (condition: breakpoint){

  //CSS Rules

}

Min-width -> from this point (starting from)

Max-width ->up to this width

Usually we define min width because we start from the smallest device to larger devices (mobile first)

@media screen and (min-width: 481px){

  //CSS Rules

}

From above code: all of our styles before this media query would be applied until we get to 481px, and then they will still be applied unless we override them. Cascading still works.

Install markdown all in one extension in vs code and then go the the .md(marked down) file and press ctrl+shft+v to see md

/\* small \*/

/\*from 576px to 576px+, greater than 576px following styles will be applied\*/

@media screen and (min-width: 576px) {

  body{

    background-color: green;

    background-image: radial-gradient(whitesmoke, green);

  }

  nav{

    display: none;

  }

}

HTML body

    <header>

        <h1>Header</h1>

        <nav><h2>Nav</h2></nav>

    </header>

    <main>

        <h2>Main</h2>

    </main>

    <footer>

        <h2>Footer</h2>

    </footer>

CSS

@import url("https://fonts.googleapis.com/css2?family=Nunito&display=swap");

/\* || RESET \*/

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

/\* || GENERAL STYLES \*/

body {

  font: 1.5rem "Nunito", sans-serif;

  min-height: 100vh;

  background-color: #475569;

  background-image: radial-gradient(whitesmoke, #475569 );

  display: flex;

  flex-direction: column;

}

header, nav, main, footer{

  display: grid;

  place-content: center;

  grid-template-columns: 100%;

  text-align: center;

}

header, footer{

  position: sticky;

  background-color: #1e293b;

  color: whitesmoke;

}

nav{

  background-color: white;

  color: black;

  padding: 0.5rem;

  border-bottom: 2px solid black;

}

header{

  top: 0;

}

main{

  flex-grow: 1;

}

footer{

  bottom: 0;

}

/\* small \*/

/\*from 576px to 576px+, greater than 576px following styles will be applied\*/

@media screen and (min-width: 576px) {

  body{

    background-color: green;

    background-image: radial-gradient(whitesmoke, green);

  }

  nav{

    display: none;

  }

}

/\* Medium \*/

/\*from 768px to 768px+, greater than 768px following styles will be applied\*/

@media screen and (min-width: 768px) {

  body {

    background-color: gold;

    background-image: radial-gradient(whitesmoke, gold);

  }

}

/\* Large \*/

/\*from 992px to 992px+, greater than 992px following styles will be applied\*/

@media screen and (min-width: 992px) {

  body {

    background-color: firebrick;

    background-image: radial-gradient(whitesmoke, firebrick);

  }

}

/\* XL \*/

/\*from 1200px to 1200px+, greater than 1200px following styles will be applied\*/

@media screen and (min-width: 1200px) {

  body {

    background-color: rebeccapurple;

    background-image: radial-gradient(whitesmoke, rebeccapurple);

  }

}

/\*mobile device landscape\*/

@media screen and (max-height: 425px) and (min-aspect-ratio: 7/4) {

  body {

    background-color: dodgerblue;

    background-image: radial-gradient(whitesmoke, dodgerblue);

  }

  h1,h2{

    font-size: 1.5rem;

  }

  nav{

    display: none;

  }

}

Lesson 18 Card Project

HTML

    <header>

      <h1>Our Staff</h1>

      <nav>

        <a href="#profile1">Joe</a>

        <a href="#profile2">Matt</a>

        <a href="#profile3">Jane</a>

      </nav>

    </header>

    <main>

      <article id="profile1" class="card">

        <figure>

          <img src="./img/profile1-500x500.png" alt="Joe Coder" width="500" height="500" loading="lazy">

          <figcaption>Joe Coder</figcaption>

        </figure>

        <!-- q stands for quote -->

        <p><q>I code stuff</q></p>

      </article>

      <article id="profile2" class="card">

        <figure>

          <img src="./img/profile2-500x500.png" alt="Matt

          Designer" width="500" height="500" loading="lazy">

          <figcaption>Matt

            Designer

          </figcaption>

        </figure>

        <p><q>I design stuff</q></p>

      </article>

      <article id="profile3" class="card">

        <figure>

          <img src="./img/profile3-500x500.png" alt="Joe Dev Rel" width="500" height="500" loading="lazy">

          <figcaption>Joe <span class="nowrap">Dev Rel</span></figcaption>

        </figure>

        <p><q>I teach stuff</q></p>

      </article>

    </main>

    <footer>

      <p>&copy Acme Co.</p>

    </footer>

CSS

@import url("https://fonts.googleapis.com/css2?family=Nunito&display=swap");

/\* || RESET \*/

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

img {

  display: block;

  max-width: 100%;

  height: auto;

}

/\*utility classes\*/

.nowrap {

  white-space: nowrap;

}

/\* || GENERAL STYLES \*/

html {

  scroll-behavior: smooth;

}

body {

  font: 1.5rem "Nunito", sans-serif;

  min-height: 100vh;

  background-color: #475569;

  background-image: radial-gradient(whitesmoke, #475569);

  display: flex;

  flex-direction: column;

}

header,

footer {

  position: sticky;

  background-color: #1e293b;

  color: whitesmoke;

  text-align: center;

}

header {

  top: 0;

}

nav {

  background-color: #fff;

  padding: 0.5rem;

  border-bottom: 2px solid #000;

  font-weight: bolder;

  display: flex;

  justify-content: space-evenly;

}

nav a,

nav a:visited {

  color: black;

}

nav a:hover,

nav a:focus {

  /\*in mobile we can see this by tabbing over links\*/

  color: hsla(0, 0%, 20%, 0.6);

}

main {

  flex-grow: 1;

  display: flex;

  flex-direction: column;

  align-items: center;

  gap: 1.5rem;

  padding: 1rem;

}

footer {

  bottom: 0;

}

/\*profile card\*/

.card {

  scroll-margin-top: 8rem;

  /\*350px is the maximum value\*/

  width: min(100%, 350px);

  background-color: #cbd5e1;

  border: 2px solid black;

  border-radius: 15px;

  padding: 1rem;

  display: flex;

  flex-direction: column;

  align-items: center;

}

.card figure {

  display: flex;

  flex-flow: column nowrap;

}

.card img {

  border: 5px double #333;

  border-radius: 50%;

}

.card figcaption {

  font-weight: bolder;

  font-size: 2rem;

  margin: 1rem;

  text-align: center;

}

/\* || SMALL \*/

@media screen and (min-width: 576px) {

  main {

    justify-content: center;

    flex-flow: row wrap;

    padding: 1rem;

  }

  .card {

    width: min(100%, 400px);

  }

  .card:last-child {

    order: -1;

  }

}

/\* || MEDIUM \*/

@media screen and (min-width: 768px) {

  nav {

    display: none;

  }

  .card {

    width: min(100%, 325px);

  }

  .card figure {

    flex-flow: column-reverse;

  }

  .card p {

    margin-top: 1rem;

  }

}

/\* || LARGE \*/

@media screen and (min-width: 992px) {

  .card {

    width: min(100%, 400px);

  }

  .card:nth-child(2) {

    order: -1;

  }

}

/\* || XL  \*/

@media screen and (min-width: 1200px) {

  .card {

    width: min(calc(33% - 1rem), 500px);

  }

}

/\* || MOBILE DEVICE LANDSCAPE  \*/

@media screen and (max-height: 425px) and (min-aspect-ratio: 7/4) {

  h1 {

    font-size: 1.5rem;

  }

  main {

    flex-flow: row nowrap;

    justify-content: space-evenly;

    align-items: stretch;

  }

  .card {

    width: min(calc(33% - 0.25rem), 200px);

  }

  .card figcaption,

  .card p {

    font-size: 1rem;

    text-align: center;

  }

  nav {

    display: none;

  }

}

Lesson 19 Pseudo Selectors (class and element)

Pseudo elements::

Pseudo classes :

nav a:active{

  color: red;

}

Font color becomes red when keep pressed

Instead of these lines

nav a:hover,

nav a:focus {

  /\*in mobile we can see this by tabbing over links\*/

  color: hsla(0, 0%, 20%, 0.6);

}

We can write dry code by adding is pseudo class

nav :is(a:hover, a:focus) {

  /\*in mobile we can see this by tabbing over links\*/

  color: hsla(0, 0%, 20%, 0.6);

}

Instead of these lines

nav a:link,

nav a:visited {

  color: black;

}

We can select both link and visited pseudo class

nav a:any-link {

  color: black;

}

In following case specificity is 0,0,1

header,

footer {

  position: sticky;

  background-color: #1e293b;

  color: whitesmoke;

  text-align: center;

}

In following case also specificity is 0,0,1

header, footer, .card {

  position: sticky;

  background-color: #1e293b;

  color: whitesmoke;

  text-align: center;

}

By adding :is specificity is now 0,1,0

:is(header, footer, .card) {

  position: sticky;

  background-color: #1e293b;

  color: whitesmoke;

  text-align: center;

}

In following case specificity is 0,0,0

:where(header, footer, .card) {

  position: sticky;

  background-color: #1e293b;

  color: whitesmoke;

  text-align: center;

}

To add a border to the target when we link via ID(whoever the target/selected)

.card:target{

  border: 3px solid rebeccapurple;

}

To select any image with alt attribute

.card img[alt]

To find which image is not added an alt attribute

.card img:not([alt]){

  border: 10px solid red;

}



To select specific card

.card:nth-child(2){

  background-color: papayawhip;

}

Order depends on the html document

Instead of we can put add/even instead of 2

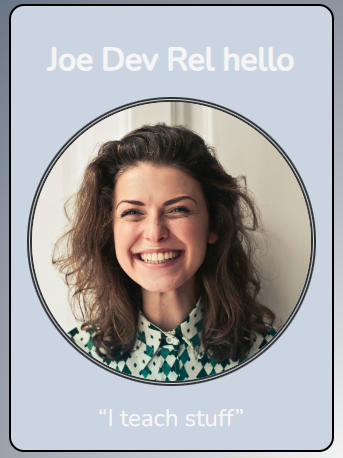
**Pseudo elements**

to add some content via css after a certain element

.card figcaption::after{

  content: ' hello';

}



We cannot select the text added via pseudo element in webpage

.card figcaption::after{

  content: '❤️';

  display: block;

}

.card p::before{

  content: open-quote;

}

.card p::after {

  content: close-quote;

}

We can style them as well

.card p::before{

  content: open-quote;

  font-size: 3em;

  position: absolute;

  top: -0.25em;

  left: -0.5em;

}

.card p::after {

  content: close-quote;

  font-size: 3em;

    position: absolute;

    top: -0.25em;

    right: -0.5em;

}



We can remove <q> elements in html document

To enlarge first letter using pseudo element

.card figcaption::first-letter{

  font-size: 3rem;

}

We can do the same for first line as well

.card figcaption::first-line{

  font-size: 3rem;

}

Everything inherit from :root. Even html

:root{

  /\*color\*/

  --BGCOLOR:#125569;

}

In any element

  background-color: var(--BGCOLOR);

Two variables can have same values or colors

We are not limited to colors, we can use

  --FF: "Nunito", sans-serif;

  --FS: 1.5rem;

  --FS-XL: 3rem;

 /\*general\*/

  --PADDING: 0.5rem;

 --SHADOWS: 0 6px -5px var(--BGCOLOR);

Ctrl+shift+L will select all occurrences.

We can create a class in html doc and we can call it in css file and within that we can reassign variable values. So that it will be applied only to related class.

To add dark theme

@media (prefers-color-scheme: dark){

  :root{

    --BGCOLOR: #000;

    --ALT-BGCOLOR: #333;

  }

}

Dark theme will only be applied if we have set color theme to dark on windows

**CSS Functions**

:root {

  /\* FONT \*/

  --FF: "Nunito", sans-serif;

  --FS: min(2.25rem, 3vh);

  --FS-XL: 3rem;

Always going to select least value from 2.25rem and 3vh.

It is necessary for one value to be absolute and another relative value

We can check the font size in computed tab in inspect tools

Absolute value will become the maximum value, font size will never exceed that value

:root {

  /\* FONT \*/

  --FF: "Nunito", sans-serif;

  --FS: max(1.75rem, 3vh);

  --FS-XL: 3rem;

Max function will choose the maximum value in given arguments.

1.75rem will be the smallest value it will get

**Clamp function**

:root {

  /\* FONT \*/

  --FF: "Nunito", sans-serif;

  /\*minimum prefered maximum\*/

  --FS: clamp(1.75rem, 3vh, 2.25rem);

  --FS-XL: 3rem;

Clamp function allows to input minimum preferred and maximum

a:hover, a:focus{

  filter: brightness(150%);

}

a:hover, a:focus{

  filter: hue-rotate(180deg)

}

Tooltips using CSS

HTML

<aside>

            <h2>Aside</h2>

            <p class="content"><span class="tooltip" data-tooltip="This is Latin">Lorem</span> ipsum dolor sit amet consectetur adipisicing elit. Incidunt, vel, quisquam mollitia corporis nesciunt laborum modi odio, praesentium magni ratione temporibus! Ab reprehenderit vel facilis! Ducimus sunt minus cum tempore.</p>

        </aside>

CSS

.tooltip{

  border-bottom: 1px dashed orange;

  position: relative;

}

.tooltip:hover::before{

  content: attr(data-tooltip);

  position: absolute;

  top: -30px;

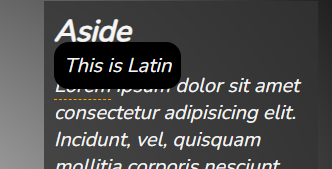
  white-space: nowrap;

  background-color: var(--DARK-COLOR);

  padding: var(--PADDING);

  border-radius: 15px;

}



Animations

To center content easily set display to grid and then place-content: center;

display: grid;

place-content: center;

Transform

div:first-child{

  background-color: dodgerblue;

  transform: translateX(50%); /\*50% is based on the size of the div\*/

}

div:nth-child(2){

  background-color: yellow;

  transform: translateY(-2rem);

}

div:last-child {

  background-color: limegreen;

  transform: translate(-50%, 2rem);/\*shorthand method for x and y\*/

}

div:first-child {

  background-color: dodgerblue;

  /\*50% is based on the size of the div\*/

  /\*transform: translateX(50%); \*/

  /\*transform: rotateX(45deg);\*/

  /\*transform: scaleX(120%);\*/

  /\*/\*transform: skewX(-10deg);\*/

}

div:nth-child(2) {

  background-color: yellow;

  /\* transform: translateY(-2rem);\*/

  /\*transform: rotateY(80deg);\*/

  /\*transform: scaleY(120%);\*/

  /\*transform: skewY(-10deg);\*/

}

div:last-child {

  background-color: limegreen;

  /\*shorthand method for x and y\*/

  /\*transform: translate(-50%, 2rem);\*/

  /\*rotate and rotateZ are same\*/

  /\*transform: rotate(30deg);\*/

  /\*transform: scale(50%, 50%);\*/

  /\* transform: skew(-10deg, 10deg);\*/

}

div:hover{

  background-color: midnightblue;

  transition: background-color;

  transition-duration: 3s;

  transition-delay: 0.5s;

}

div:last-child:hover{

  transform: rotate(180deg);

  transition: transform;

  transition-duration: 3s;

  transition-delay: 0.5s;

}

Or

div:hover{

  background-color: midnightblue;

  transition: background-color, transform;

  transition-duration: 2s, 3s;

  transition-delay: 0.5s;

}

div:last-child:hover{

  transform: rotate(180deg);

}

With shorthand method

div:hover{

  background-color: midnightblue;

  transition: all 2s 0.5s;

  transition-timing-function: ease-out;

}

div:last-child:hover{

  transform: rotate(180deg);

}

Ease is the default transition timing function

Animation basic

.animate {

  animation-name: slide;

  animation-duration: 5s;

  animation-timing-function: ease-in-out;

  animation-delay: 1s;

  animation-iteration-count: 5;

}

@keyframes slide {

  0% {

    transform: translateX(0);

  }

  33% {

    transform: translateX(300px) rotate(180deg);

  }

  66% {

    transform: translateX(-300px) rotate(-180deg);

  }

  0% {

    transform: translateX(0);

    background-color: rebeccapurple;

  }

}

With animation-direction and animation-fill-mode

.animate {

  animation-name: slide;

  animation-duration: 5s;

  animation-timing-function: ease-in-out;

  animation-delay: 1s;

  animation-iteration-count: 2;

  animation-direction: normal;/\*first goes to forward and then backward alternatively\*/

  animation-fill-mode: forwards;/\*ending state\*/

}

@keyframes slide {

  0% {

    transform: translateX(0);

  }

  33% {

    transform: translateX(300px) rotate(180deg);

  }

  66% {

    transform: translateX(-300px) rotate(-180deg);

  }

  100% {

    transform: translateX(0);

    background-color: rebeccapurple;

  }

}

shorthand method

animation: name duration timing-function delay iteration-count direction fill-mode;

code with short hand method

.animate {

  animation: slide 5s ease 1s 2 normal forwards;

}

@keyframes slide {

  0% {

    transform: translateX(0);

  }

  33% {

    transform: translateX(300px) rotate(180deg);

  }

  66% {

    transform: translateX(-300px) rotate(-180deg);

  }

  100% {

    transform: translateX(0);

    background-color: rebeccapurple;

  }

}