Table Less Layout design

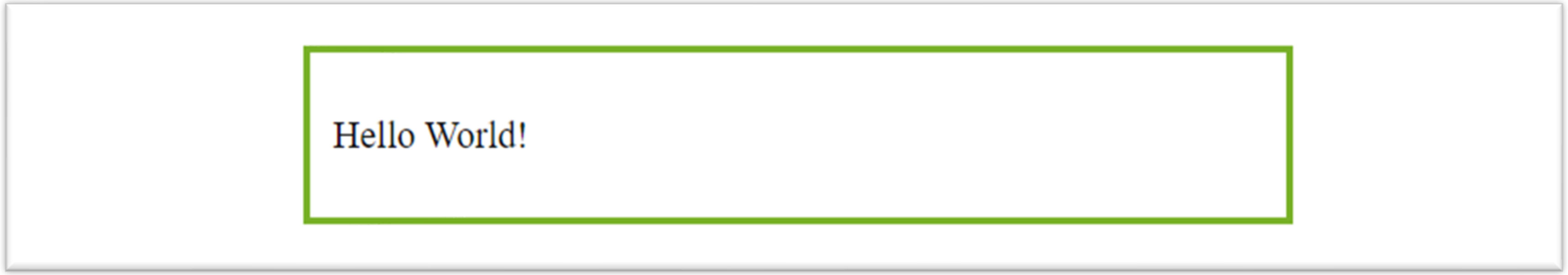
ALIGNMENT FLOATING POSITIONING FLEX-BOX

Design Your Layout using

- Alignment
- Float
- Position
- Flex

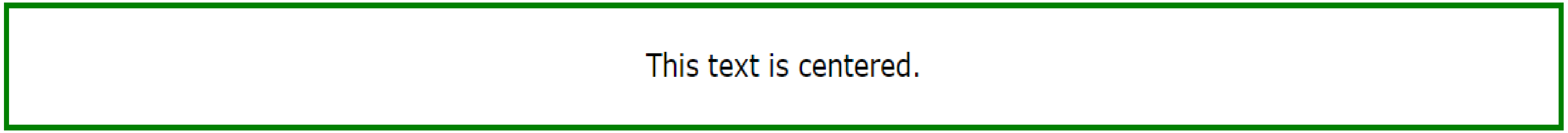
Auto Margin

```
.center {
  margin: auto;
  width: 60%;
  border: 3px solid #73AD21;
  padding: 10px;
}
```


Hello World!

text-align: center;

```
.center {
  text-align: center;
  border: 3px solid green;
}
```



This text is centered.

float: right;

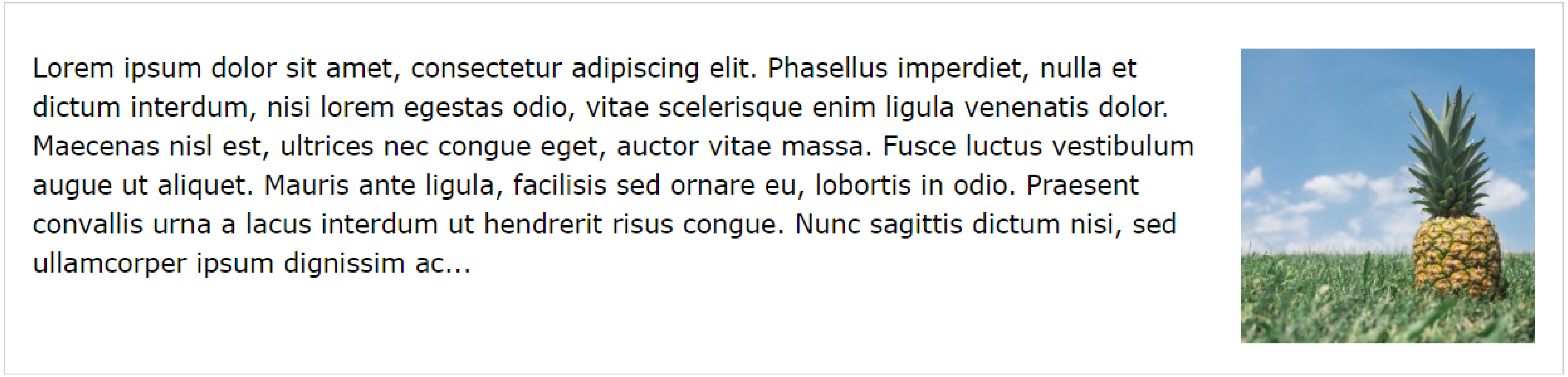
```
img {
  float: right;
}
```

left - The element floats to the left of its container

right - The element floats to the right of its container

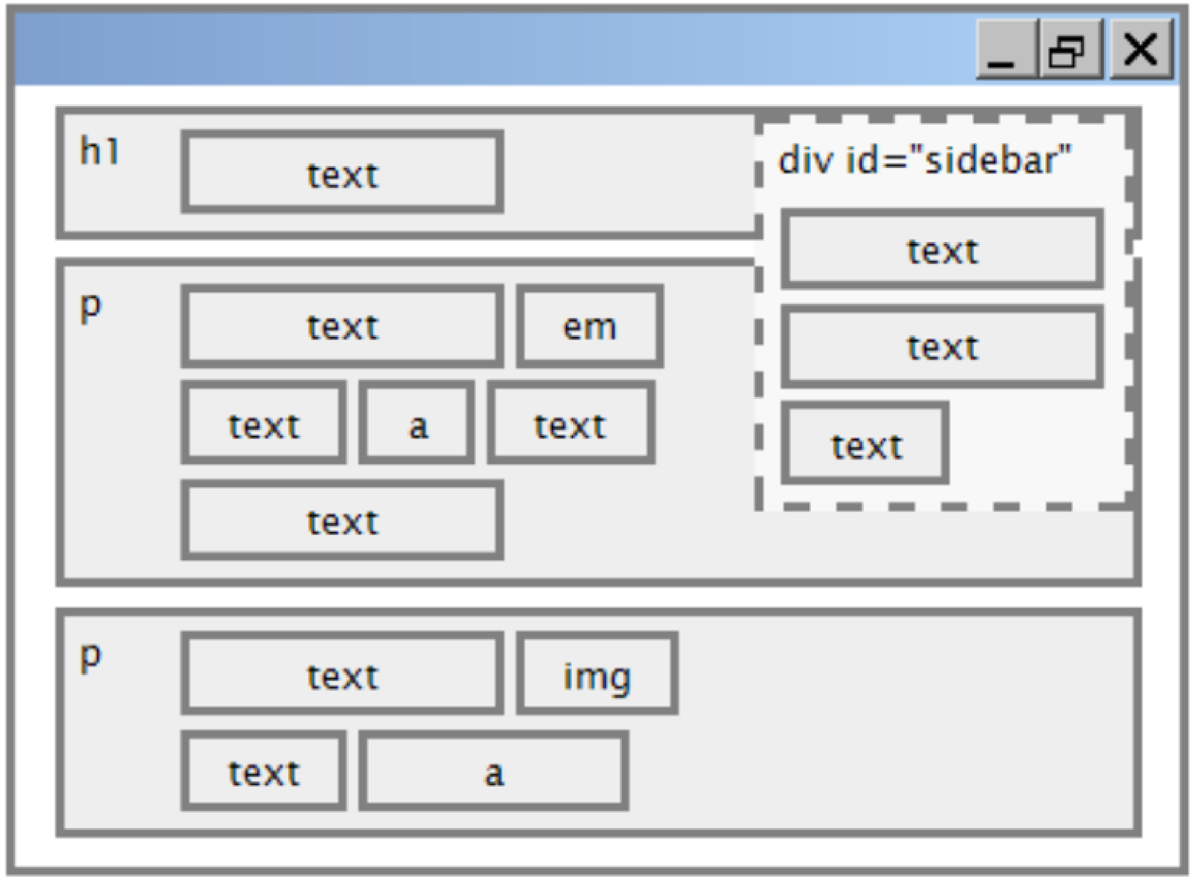
none - The element does not float (will be displayed just where it occurs in the text). This is default

inherit - The element inherits the float value of its parent



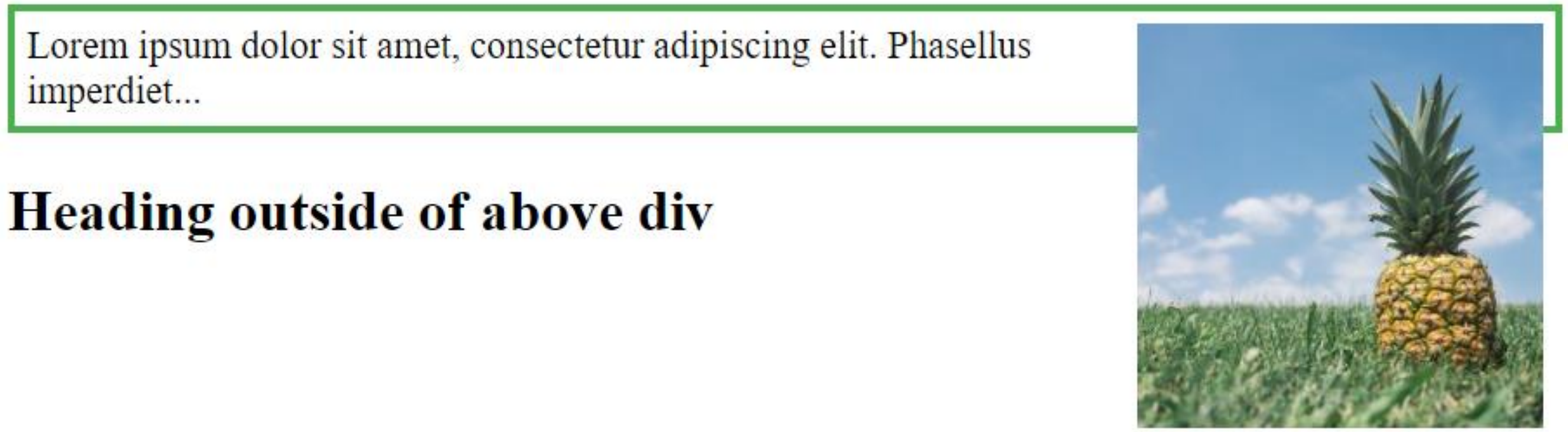
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...

Floating sidebar



Overflow

Floating element overflows the container.

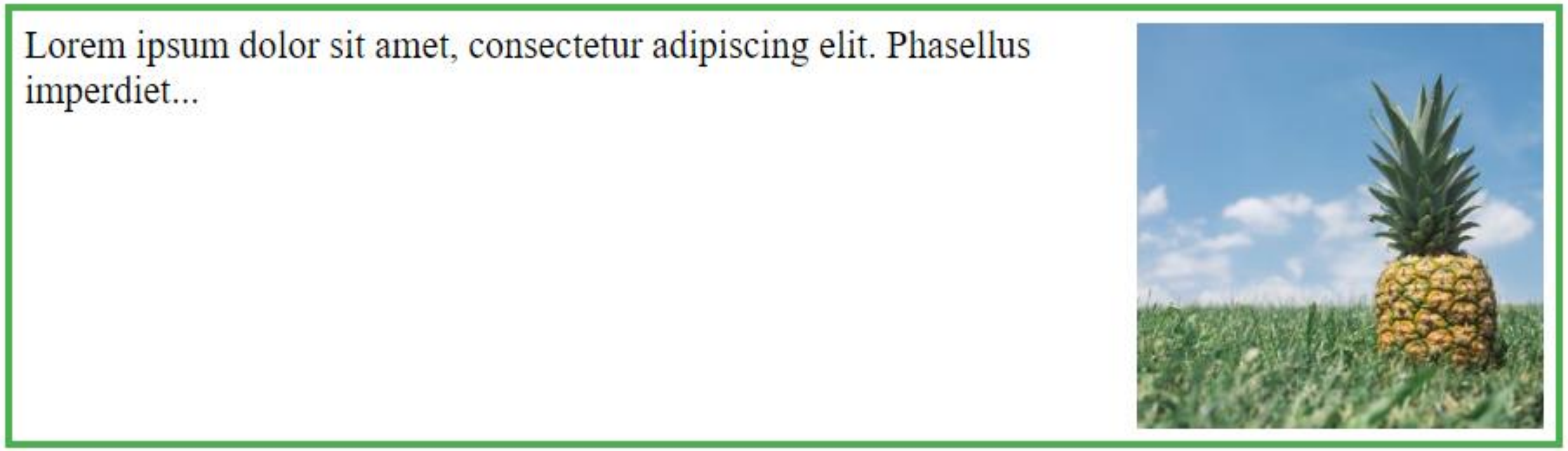


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...

Heading outside of above div

overflow:hidden

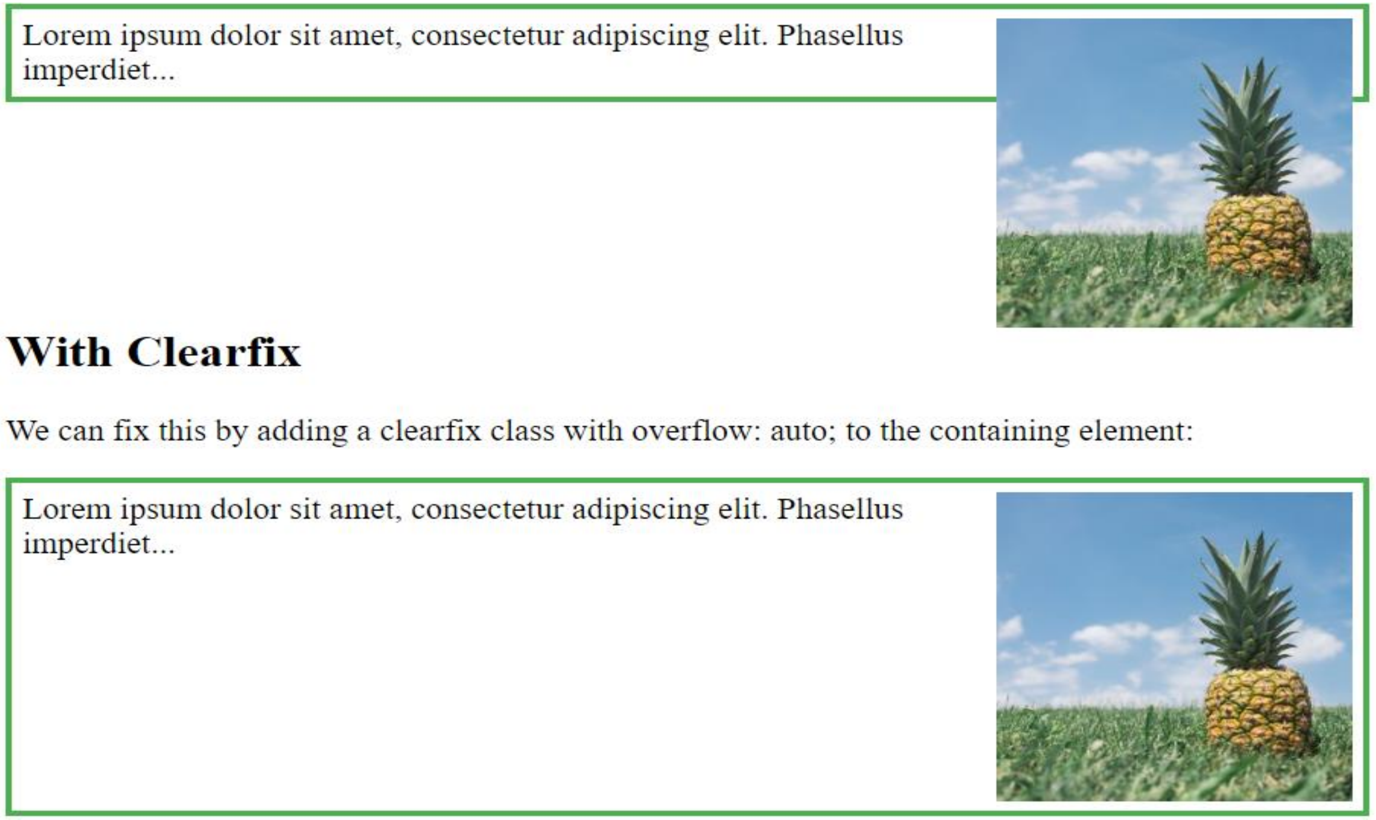
We can fix this by adding a overflow: hidden; to the container element:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...

clear:right
can be left,both or none also

```
<h2 style="clear:right">With Clearfix</h2>
```



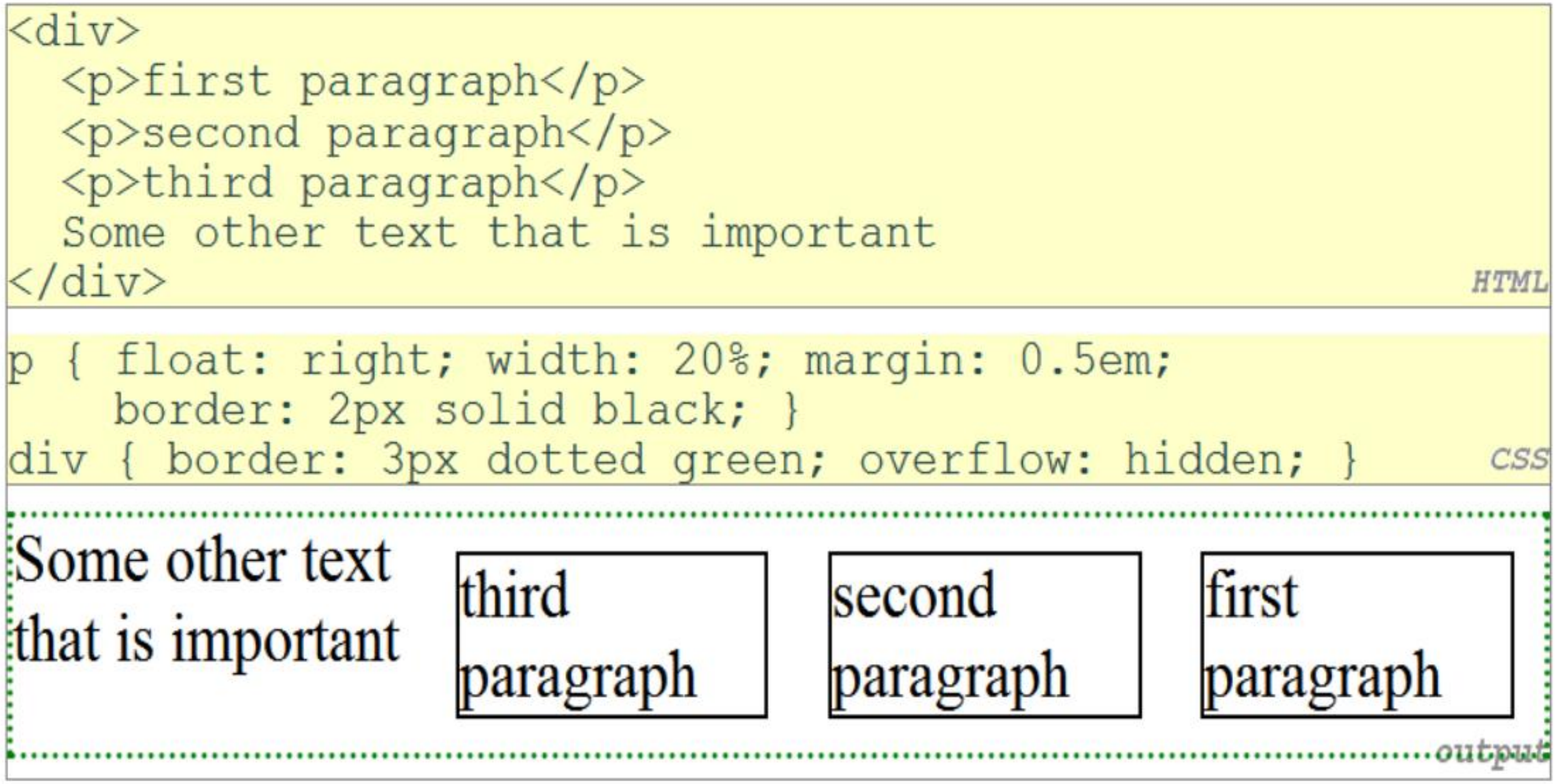
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...

With Clearfix

We can fix this by adding a clearfix class with overflow: auto; to the containing element:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...

Multi Column Layout



```html
<div>
  <p>first paragraph</p>
  <p>second paragraph</p>
  <p>third paragraph</p>
  Some other text that is important
</div>
```

```css
p { float: right; width: 20%; margin: 0.5em;
    border: 2px solid black; }
div { border: 3px dotted green; overflow: hidden; }
```

```
Some other text      third        second       first
that is important    paragraph    paragraph    paragraph
```

Images Side By Side

```
.img-container {
  float: left;
  width: 33.33%; /* three containers (use 25% for four, and 50% for two, etc) */
  padding: 5px; /* if you want space between the images */
}
```

CSS Positioning

STILL CANT MAKE YOUR DESIGN? SWITCH TO POSITIONING

position: static;

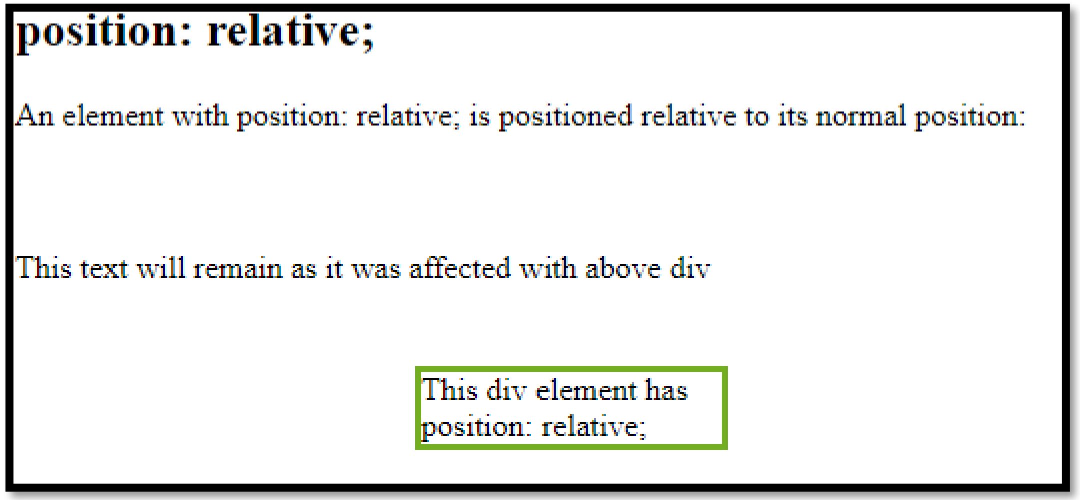
HTML elements are positioned static by default.

An element with position: static; is not positioned in any special way;

position: relative;

An element with position: relative; is positioned relative to its normal position:

```
div.relative {
  position: relative;
  left: 200px;
  top:100px;
  width: 150px;
  border: 3px solid #73AD21;
}
```



position: relative;

An element with position: relative; is positioned relative to its normal position:

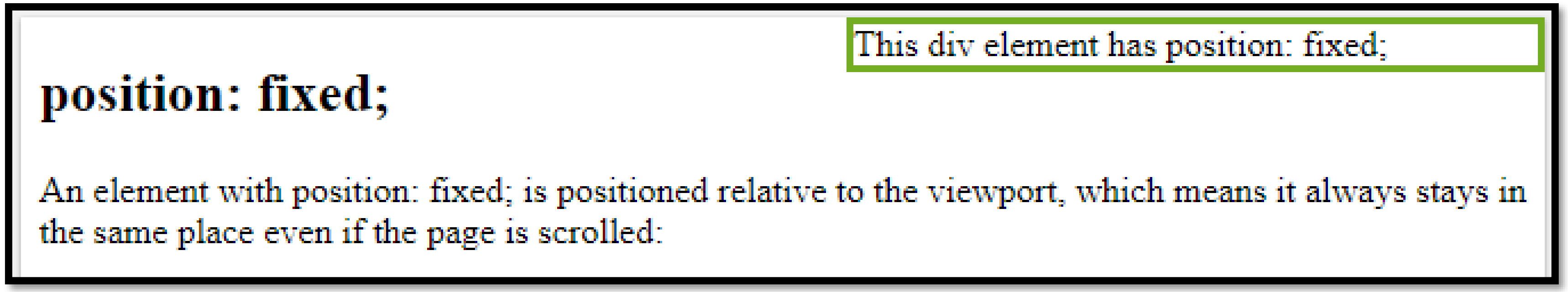
This text will remain as it was affected with above div

This div element has position: relative;

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:

```
div.fixed {
  position: fixed;
  top: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```



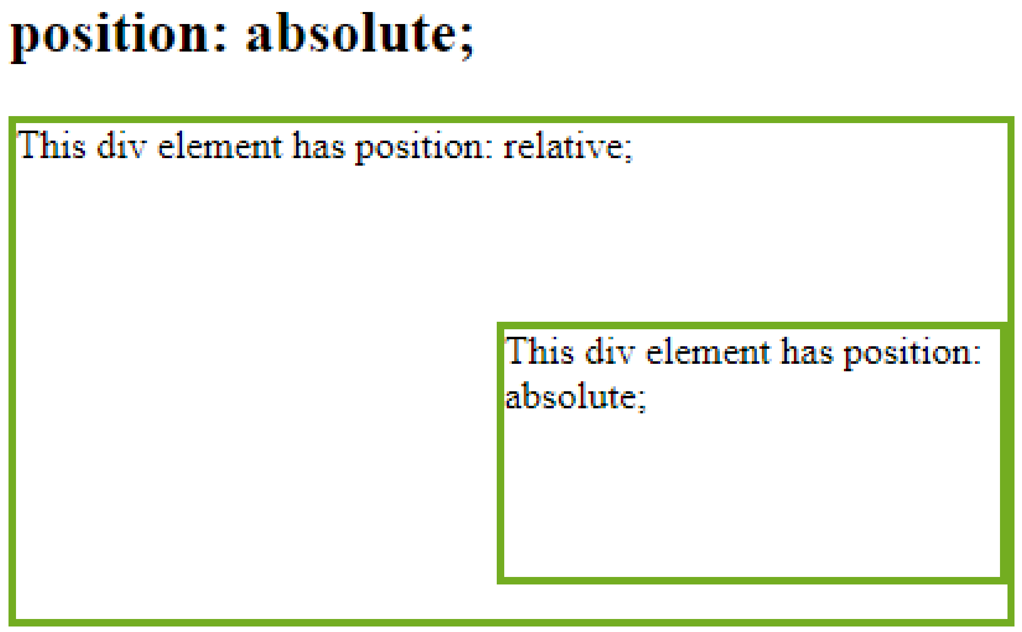
This div element has position: fixed;

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:

position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).


position: absolute;

This div element has position: relative;

This div element has position: absolute;

position: absolute;

```
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
```

```
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

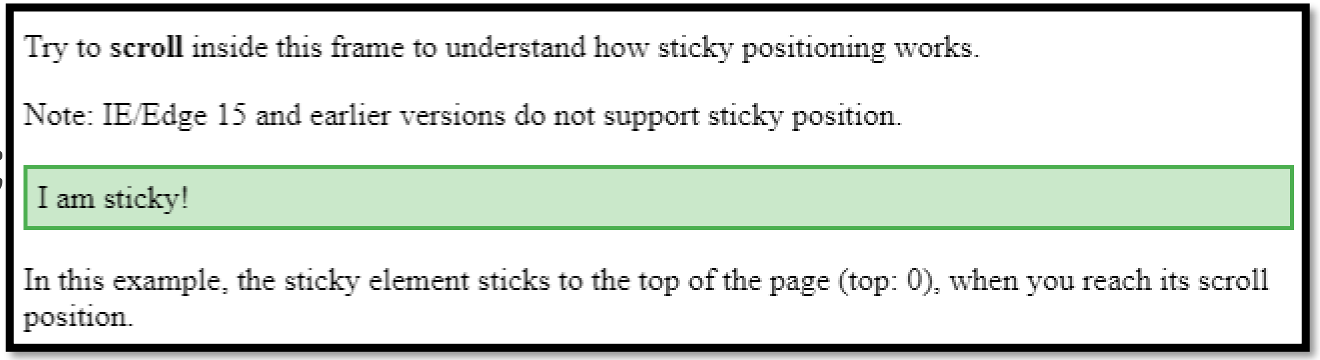
position: sticky;

An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

position: sticky;

```
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
```

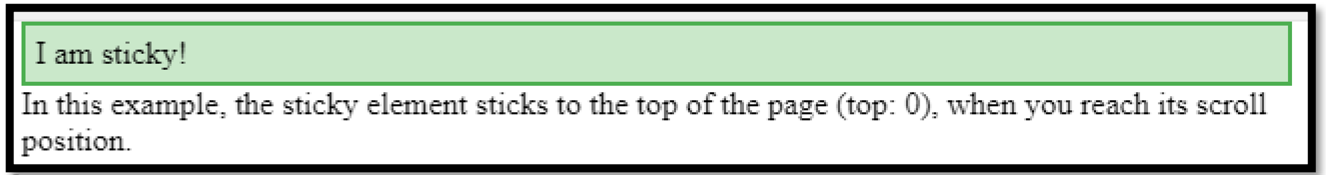


Try to **scroll** inside this frame to understand how sticky positioning works.

Note: IE/Edge 15 and earlier versions do not support sticky position.

I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.



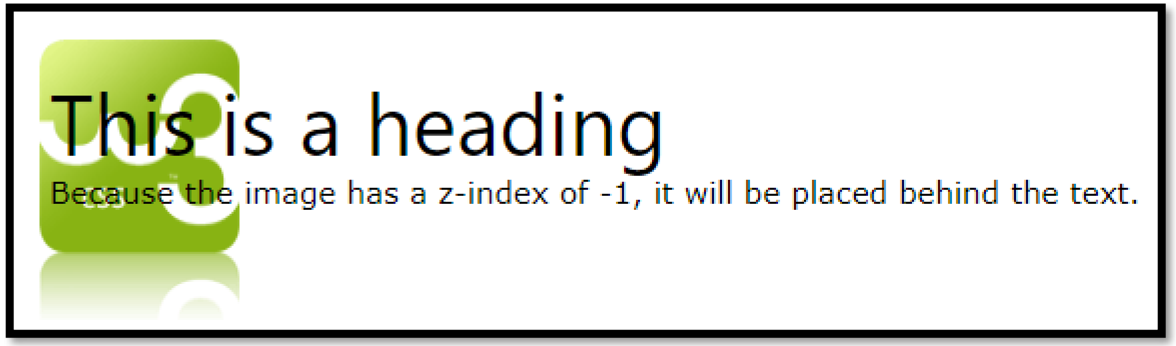
I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

Overlapping Elements

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).



This is a heading

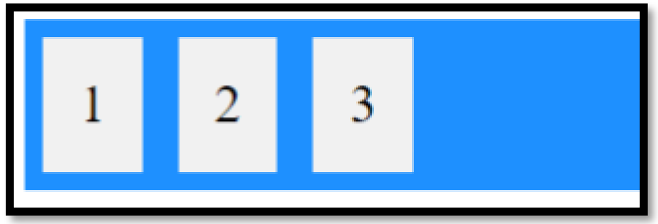
Because the image has a z-index of -1, it will be placed behind the text.

CSS Flexbox

THE BOSS

Flex Container

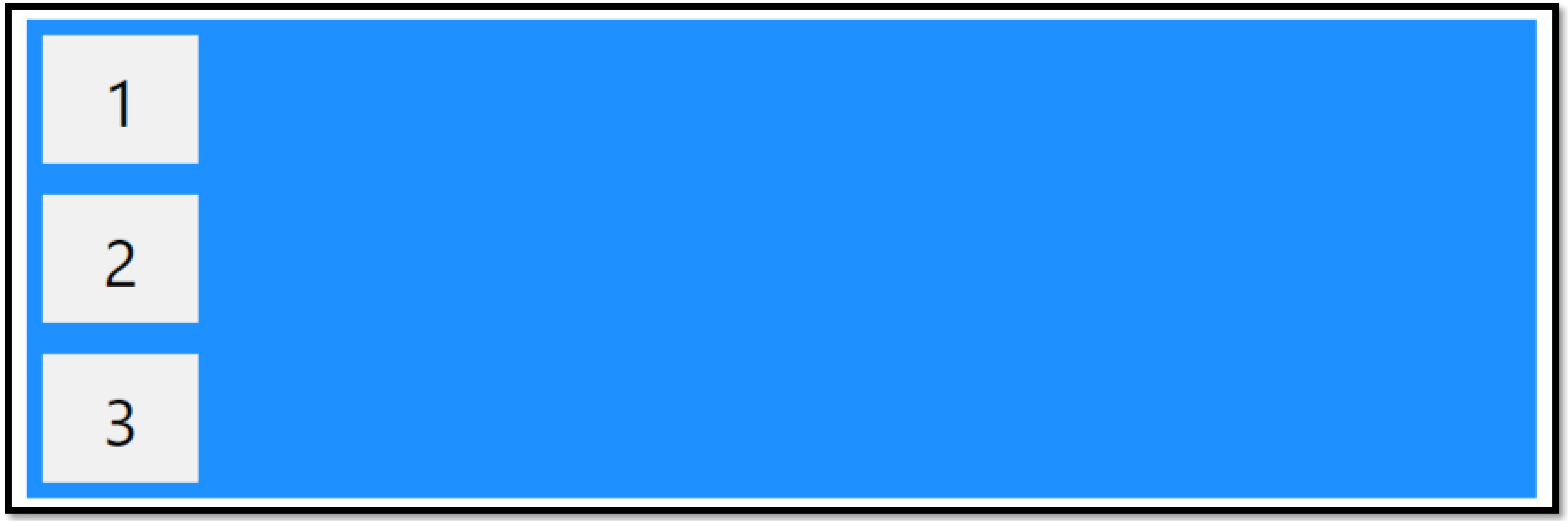
```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```



```
.flex-container {
  display: flex;
  background-color: DodgerBlue;
}
.flex-container > div {
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
```

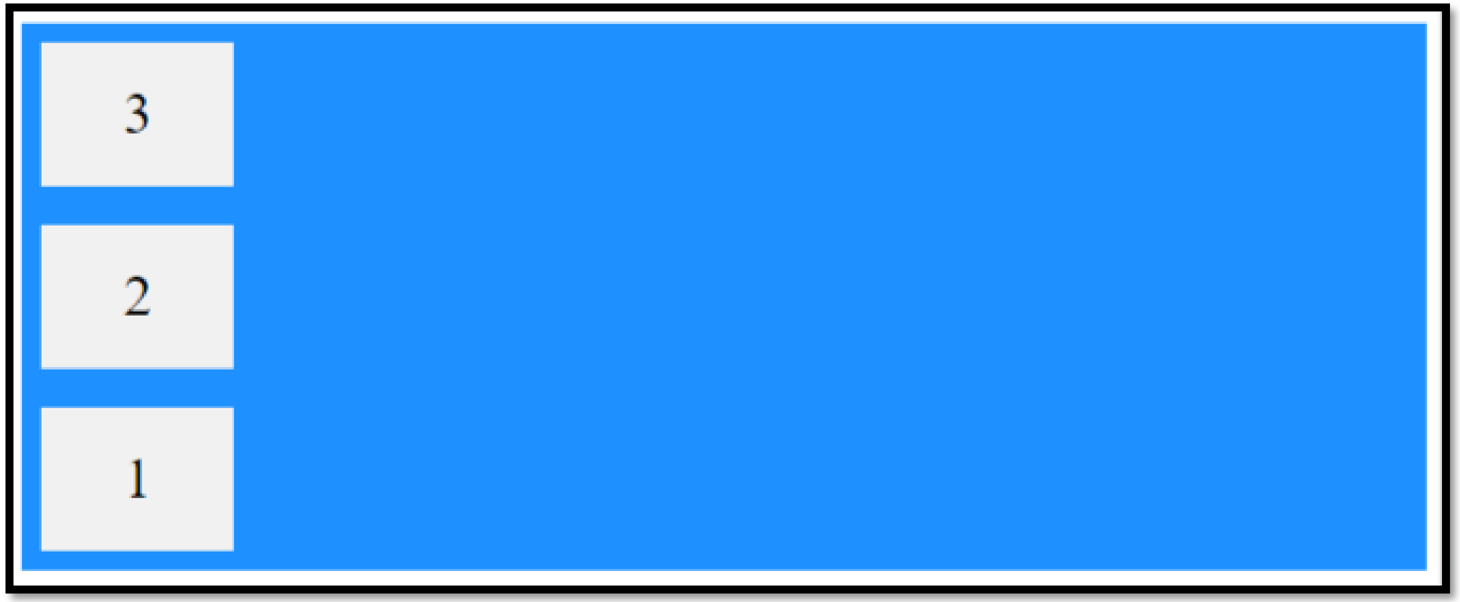
flex-direction: column;

```
.flex-container {
  display: flex;
  flex-direction: column;
}
```



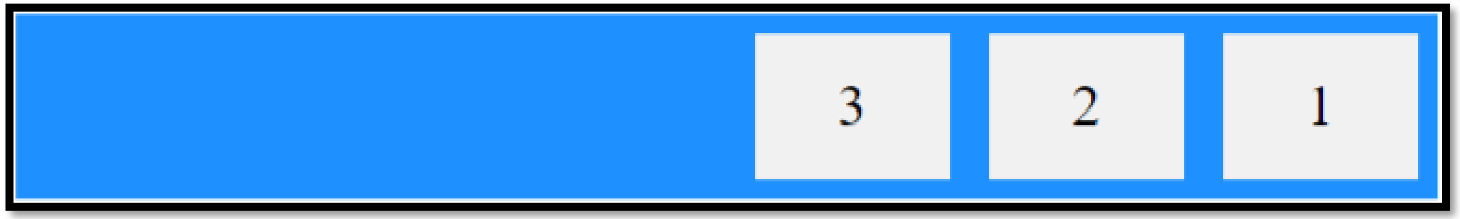
flex-direction: column-reverse;

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



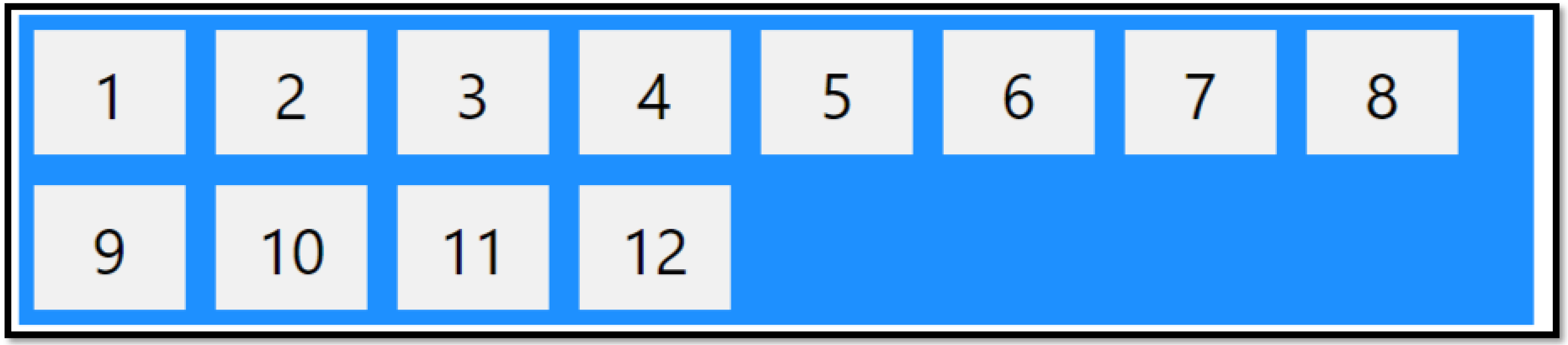
flex-direction: row-reverse;

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



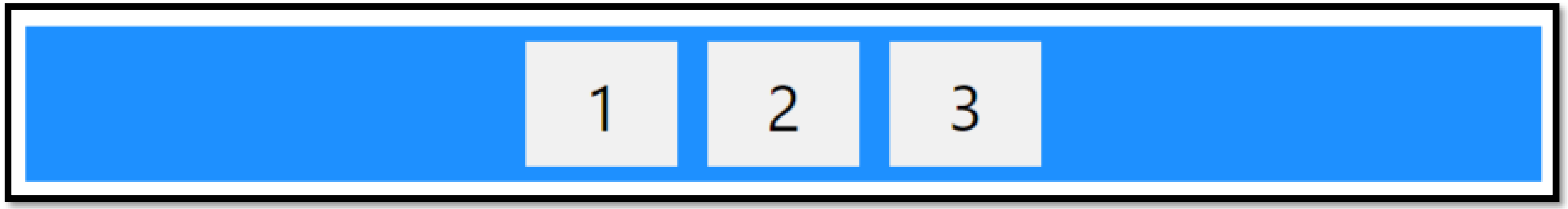
The flex-wrap Property (wrap, nowrap)

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



The justify-content Property

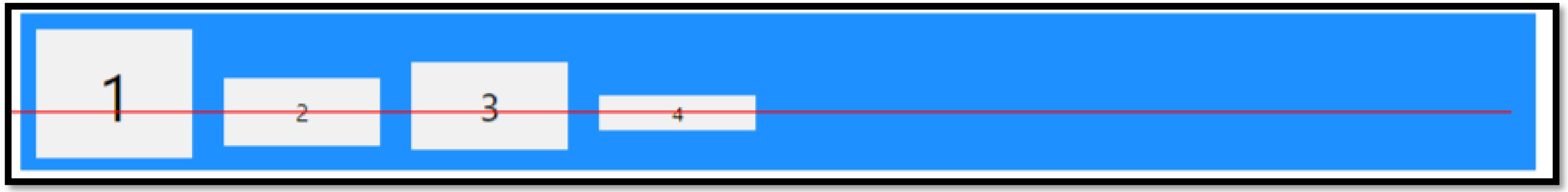
```
.flex-container {
  display: flex;
  justify-content: center;
}
```



Other Options flex-start, flex-end, space-around, space-between

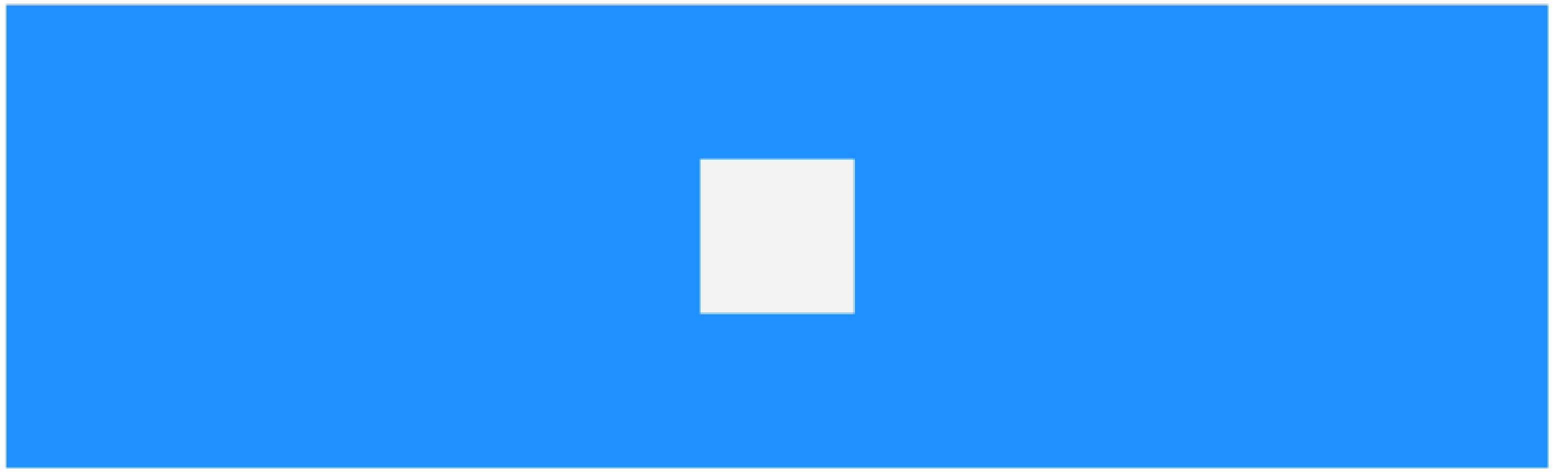
The align-items Property

```
.flex-container {
  display: flex;
  height: 200px;
  align-items: baseline;
}
```



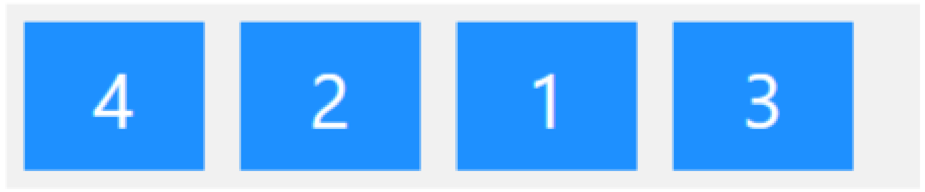
Perfect Centering

```
.flex-container {
  display: flex;
  height: 300px;
  justify-content: center;
  align-items: center;
}
```



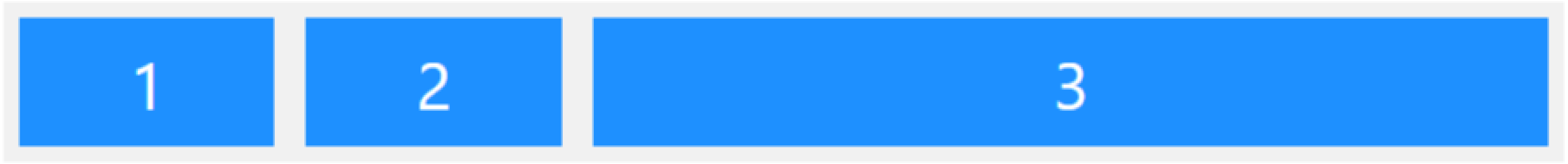
The Child Order

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```



The Flex Grow

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```

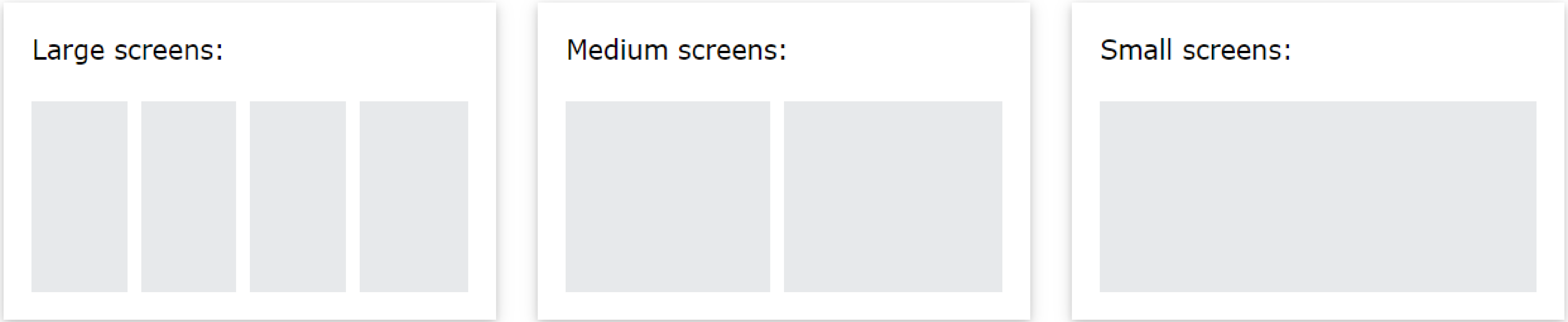


Responsiveness

```
@media screen and (min-width: 480px) {
  #leftsidebar {width: 200px; float: left;}
  #main {margin-left: 216px;}
}
```

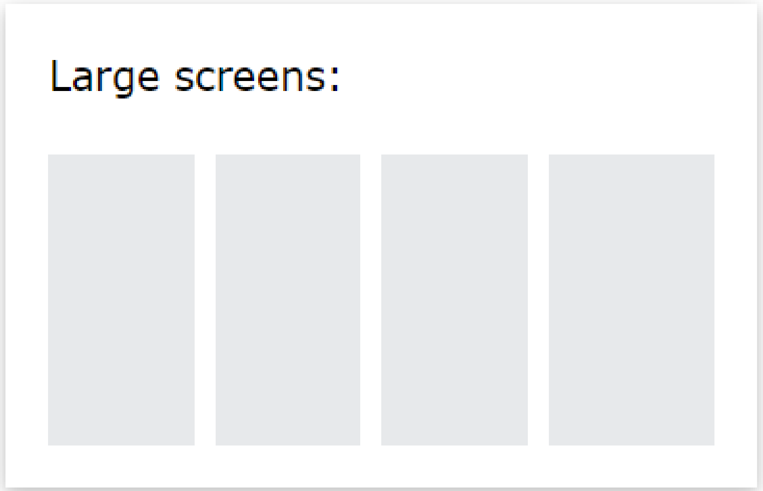
Media Queries For Columns

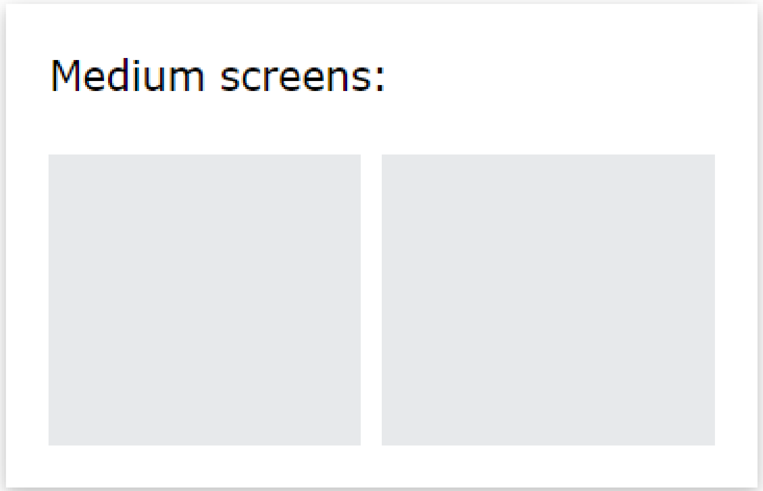
```
/* Create four equal columns that floats next to each other */
.column {
  float: left;
  width: 25%;
}
```

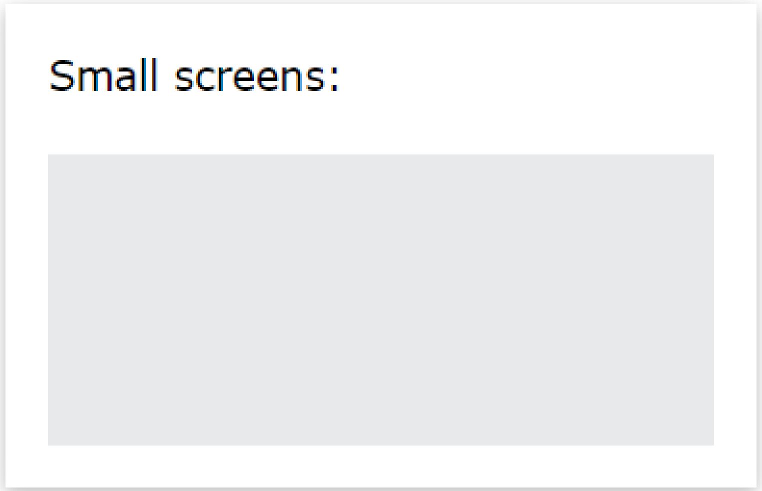


Media Queries For Columns

```
/* On screens that are 992px wide or less, go from four columns to two columns */
@media screen and (max-width: 992px) {
  .column {
    width: 50%;
  }
}
```

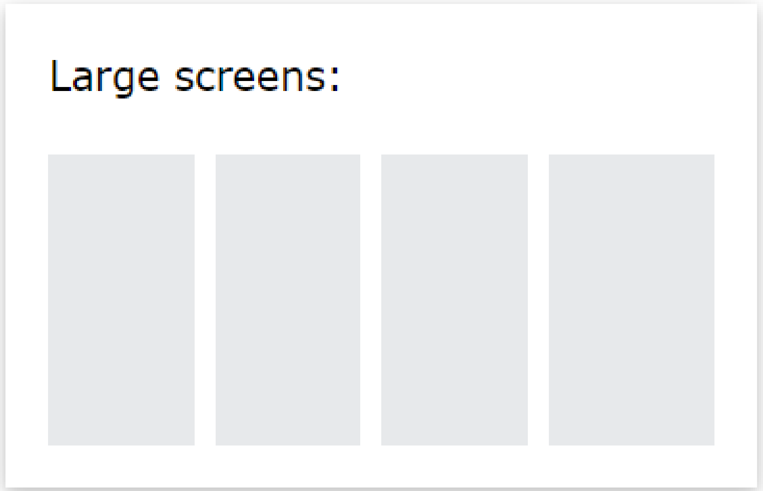


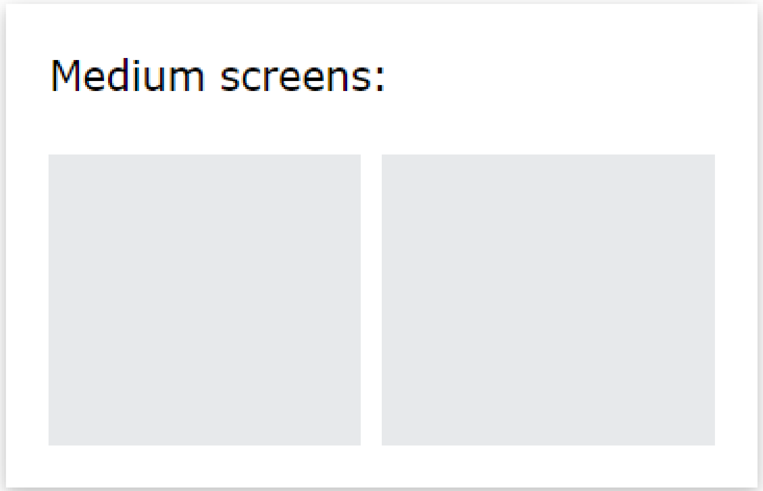


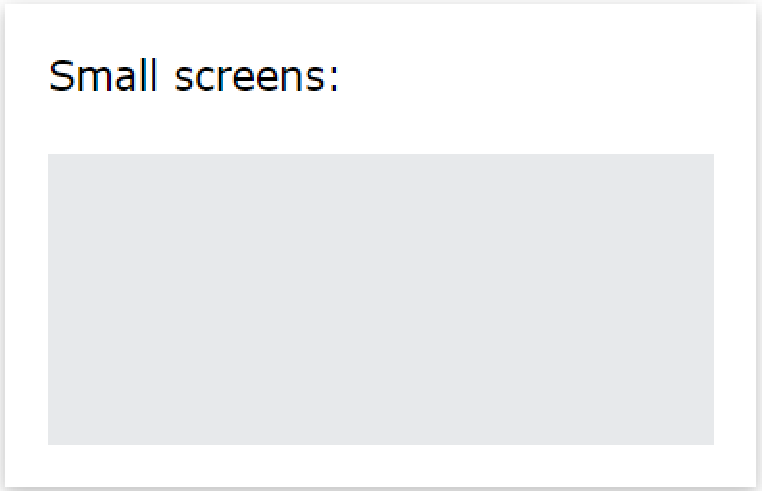


Media Queries For Columns

```
/* On screens that are 600px wide or less, make the columns stack on top of each
other instead of next to each other */
@media screen and (max-width: 600px) {
  .column {
    width: 100%;
  }
}
```



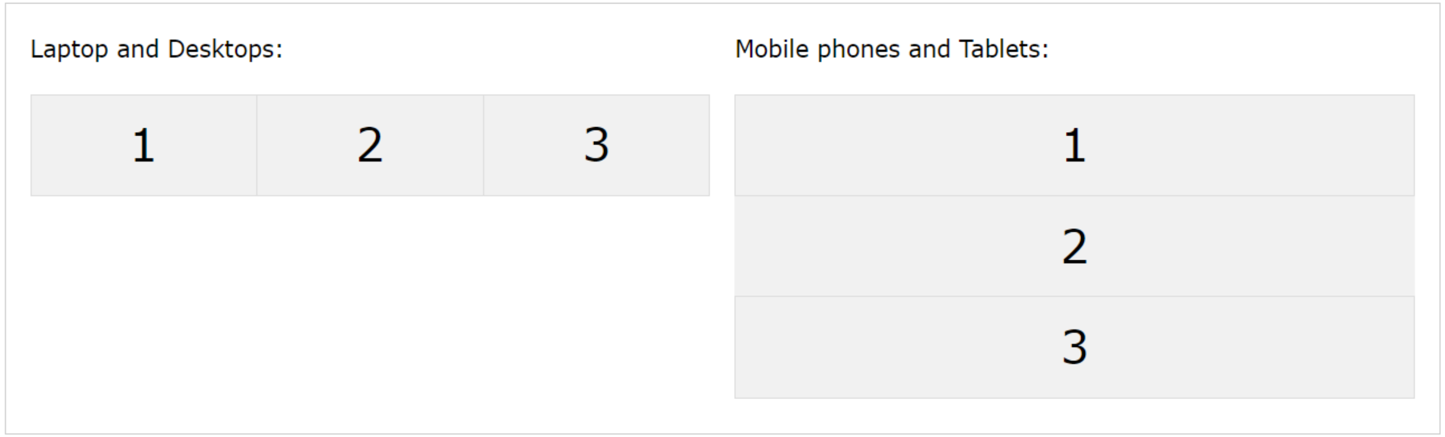




Responsive Flex

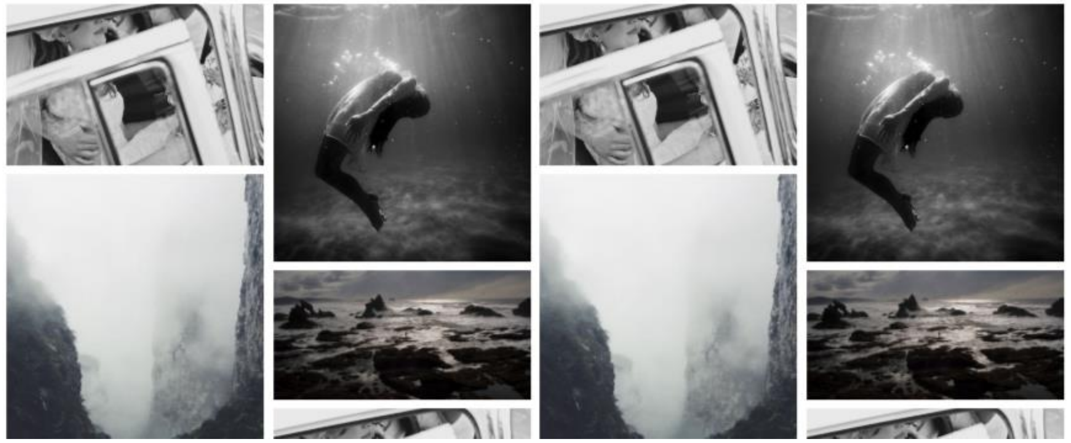
```
.flex-container {
  display: flex;
  flex-direction: row;
}

/* Responsive layout - makes a one column layout instead of a two-column layout
*/
@media (max-width: 800px) {
  .flex-container {
    flex-direction: column;
  }
}
```



Try it yourself

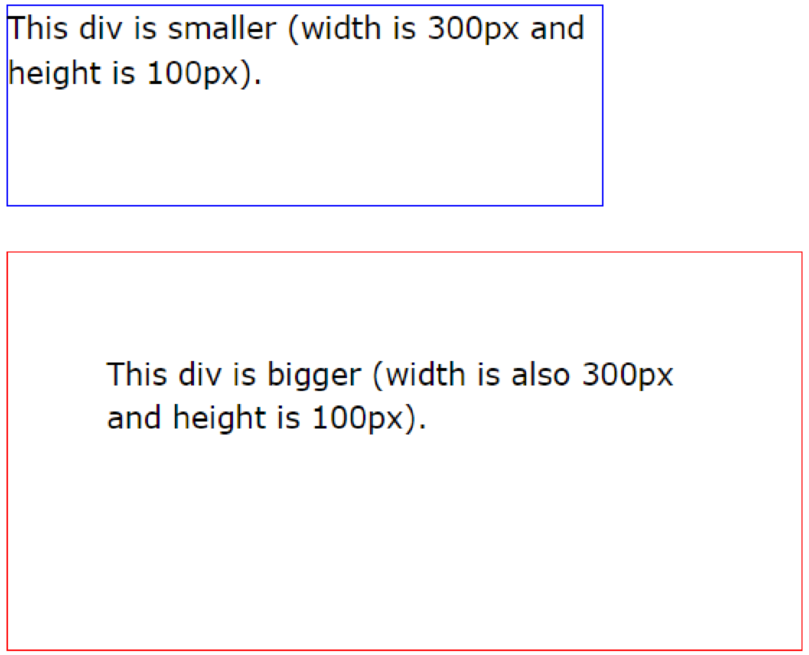
https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_image_gallery



CSS Box Sizing Issue

width + padding + border = actual width of an element

height + padding + border = actual height of an element

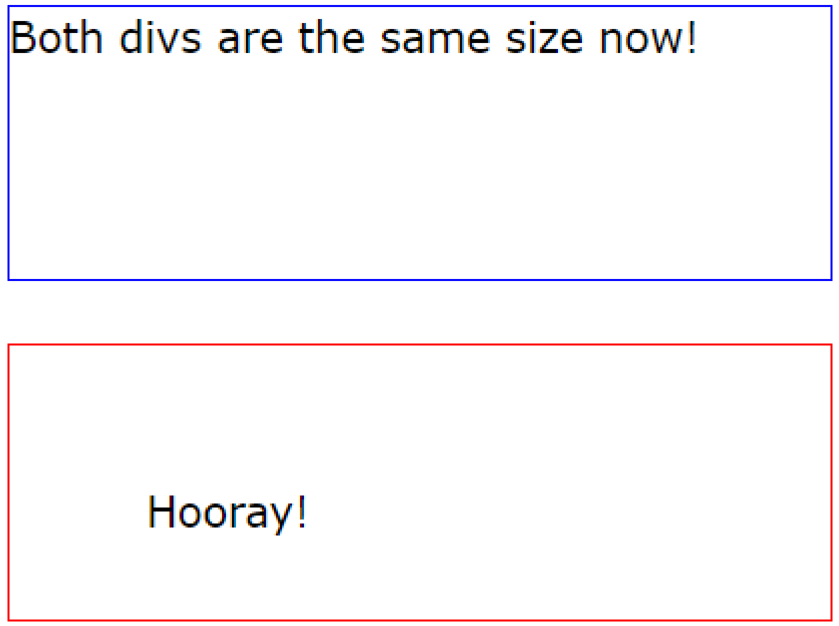


This div is smaller (width is 300px and height is 100px).

This div is bigger (width is also 300px and height is 100px).

box-sizing: border-box;

```
* {
  box-sizing: border-box;
}
```


Both divs are the same size now!

Hooray!

Animations

An animation lets an element gradually change from one style to another.

To use CSS animation, you must first specify some keyframes for the animation.

The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

```
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

Animations

```
/* Safari 4.0 - 8.0 */
@-webkit-keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* Standard syntax */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

Simple Example

THE ANIMATION CODE

```
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

HE ELEMENT TO APPLY THE ANIMATION TO

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

Use % to say what to do when

```
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}
```

Control Animation

```
  animation-name: example;
  animation-duration: 4s; /*Speed*/
  animation-delay: 2s; /*delay*/
animation-iteration-count: 3;
animation-direction: reverse;
```

Or

```
animation-iteration-count: infinite;
```

Speed Curve of the Animation

```
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
```

animation: example 5s linear 2s infinite alternate;

```
div {
  animation-name: example;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

Some Cool CSS Animations

Examples

Cool

More Cool Examples

Best Ones

Use Less for

Variables

Dynamically calculated values

Mixins

Functions

Add LESS in HTML (Slow: Not Recommended)

```
<link rel="stylesheet/less" type="text/css" href="styles.less" />

<script src="//cdnjs.cloudflare.com/ajax/libs/less.js/3.9.0/less.min.js" ></script>
```

LESS Recommended Way

Install Node From Here

Then use following commands

```
npm install -g less
 lessc styles.less styles.css
```

LESS Variables

LESS

```
@background-color: #ffffff;
@text-color: #1A237E;
p{
  background-color: @background-color;
  color: @text-color;
  padding: 15px;
}
```

CSS

```
p{
  background-color: #ffffff;
  color: #1A237E;
  padding: 15px;
}
```

LESS Mixins

LESS

```
#circle{
  background-color: #4CAF50;
  border-radius: 100%;
}
#small-circle{
  width: 50px;
  height: 50px;
  #circle
}
```

CSS

```
#circle {
  background-color: #4CAF50;
  border-radius: 100%;
}
#small-circle {
  width: 50px;
  height: 50px;
  background-color: #4CAF50;
  border-radius: 100%;
}
```

LESS Mixins With Parameters

LESS

```
#circle(@size: 25px){
  background-color: #4CAF50;
  border-radius: 100%;
  width: @size;
  height: @size;
}
#big-circle{
  #circle(100px)
}
```

CSS

```
#big-circle {
  background-color: #4CAF50;
  border-radius: 100%;
  width: 100px;
  height: 100px;
}
```

Nesting And Scope

LESS

```
ul{
  background-color: #03A9F4;
  padding: 10px;
  list-style: none;
  li{
    background-color: #fff;
    border-radius: 3px;
    margin: 10px 0;
  }
}
```

CSS

```
ul{
  background-color: #03A9F4;
  padding: 10px;
  list-style: none;
}
ul li{
    background-color: #fff;
    border-radius: 3px;
    margin: 10px 0;
  }
```

Operations

LESS

```
@div-width: 100px;
@color: #03A9F4;
#right{
  width: @div-width * 2;
  background-color: @color;
}
```

CSS

```
#right {
  width: 200px;
  background-color: #03a9f4;
}
```

LESS Functions

LESS

```
@var: #004590;
div{
  height: 50px;
  width: 50px;
  background-color: @var;
  &:hover{
    background-color: fadeout(@var, 50%)
  }
}
```

CSS

```
div {
  height: 50px;
  width: 50px;
  background-color: #004590;
}
div:hover {
  background-color: rgba(0, 69, 144, 0.5);
}
```