

# CSS Flexbox

---

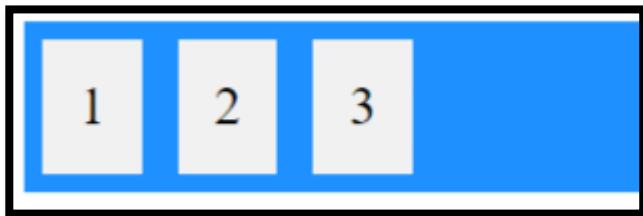
CREATE SOME AWESOMENESS

A solid orange horizontal bar at the bottom of the page.

# 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;

---

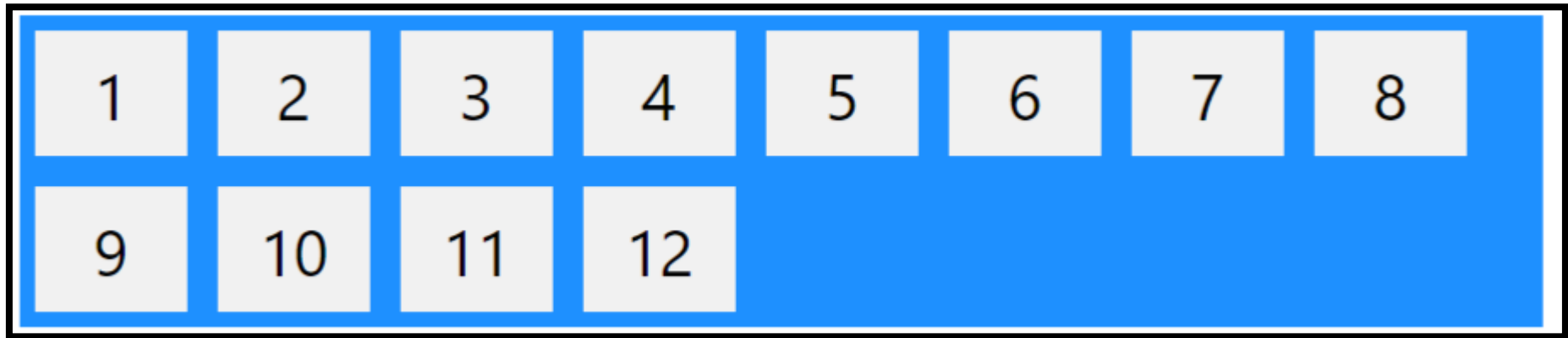
```
.flex-container {  
  display: flex;  
  flex-direction: row-reverse;  
}
```



# The flex-wrap Property (wrap, nowrap)

---

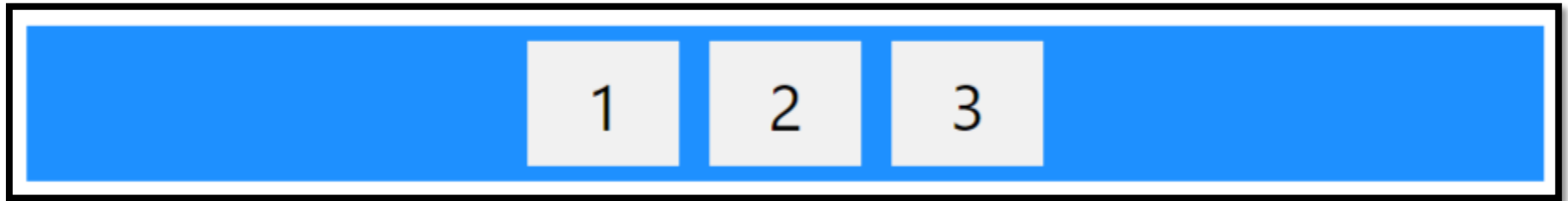
```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
}
```



# The justify-content Property

---

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

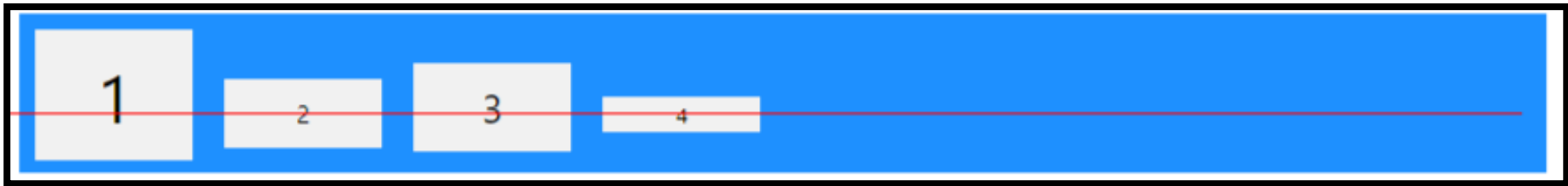


Other Options flex-start, flex-end, space-around, space-between

# 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

---

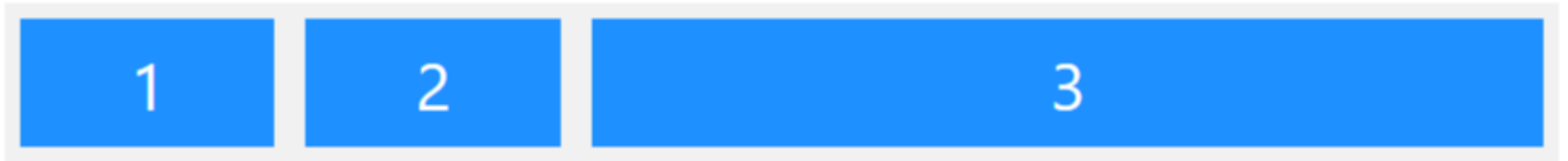
```
<div class="flex-container">  
  <div style="order: 3">1</div>  
  <div style="order: 2">2</div>  
  <div style="order: 4">3</div>  
  <div style="order: 1">4</div>  
</div>
```



# 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;}  
}
```

Property	Description
<a href="#">display</a>	Specifies the type of box used for an HTML element
<a href="#">flex-direction</a>	Specifies the direction of the flexible items inside a flex container
<a href="#">justify-content</a>	Horizontally aligns the flex items when the items do not use all available space on the main-axis
<a href="#">align-items</a>	Vertically aligns the flex items when the items do not use all available space on the cross-axis
<a href="#">flex-wrap</a>	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
<a href="#">align-content</a>	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
<a href="#">flex-flow</a>	A shorthand property for flex-direction and flex-wrap
<a href="#">order</a>	Specifies the order of a flexible item relative to the rest of the flex items inside the same container
<a href="#">align-self</a>	Used on flex items. Overrides the container's align-items property
<a href="#">flex</a>	A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties

# The flex-shrink Property

---

```
<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div style="flex-shrink: 0">3</div>  
  <div>4</div>  
  <div>5</div>  
  <div>6</div>  
  <div>7</div>  
  <div>8</div>  
  <div>9</div>  
  <div>10</div>  
</div>
```



# Try it yourself

---

[https://www.w3schools.com/css/tryit.asp?filename=trycss3\\_flexbox\\_image\\_gallery](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_image_gallery)



# CSS Positioning

---

CREATE SOME AWESOMENESS





# position: static;

---

HTML elements are positioned static by default.

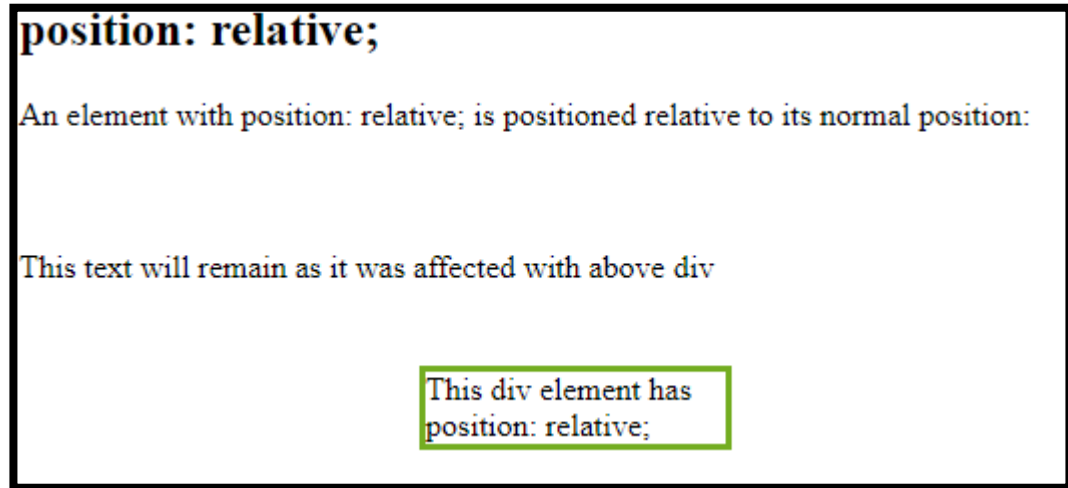
An element with position: static; is not positioned in any special way;

# 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;  
}
```

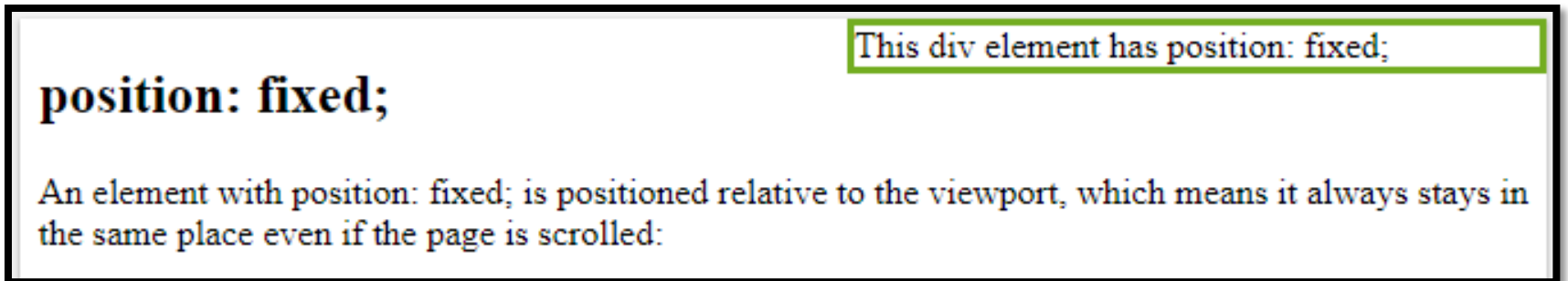


# position: fixed;

---

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;  
}
```

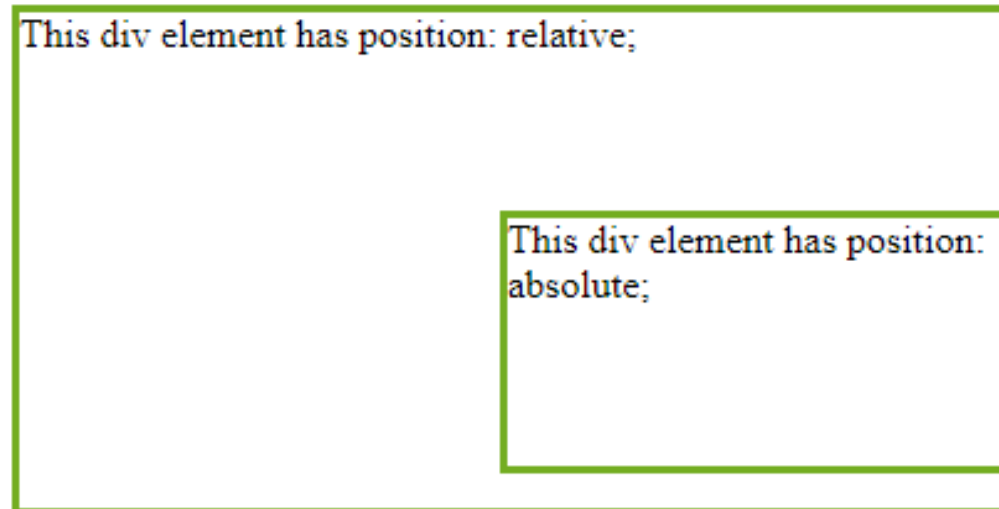


# position: absolute;

---

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

**`position: absolute;`**



# 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!

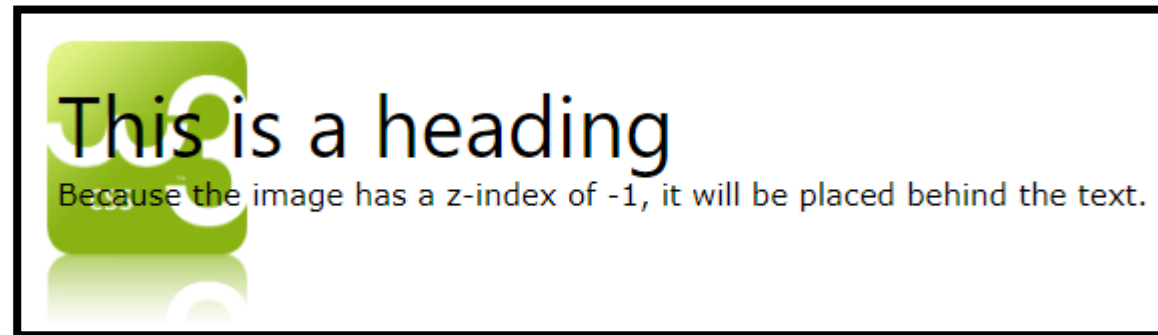
In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

# 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).





# 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;}  
}
```

## THE 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;  
}
```



# 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);
}
```