



Master React 18 and build scalable apps with a complete React learning path.

[Mozilla ads](#)

[Don't want to see ads?](#)

Using CSS transforms

By modifying the coordinate space, CSS transforms change the shape and position of the affected content without disrupting the normal document flow. This guide provides an introduction to using transforms.

CSS transforms are implemented using a set of CSS properties that let you apply affine linear transformations to HTML elements. These transformations include rotation, skewing, scaling, and translation both in the plane and in the 3D space.



Warning: Only transformable elements can be `transformed`; that is, all elements whose layout is governed by the CSS box model except for: non-replaced inline boxes, table-column boxes, and table-column-group boxes.

CSS transforms properties

Two major properties are used to define CSS transforms: `transform` (or the individual `translate`, `rotate`, and `scale` properties) and `transform-origin`.

`transform-origin`

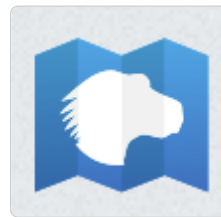
Specifies the position of the origin. By default, it is at the center of the element and can be moved. It is used by several transforms, like rotations, scaling or skewing, that need a specific point as a parameter.

[transform](#)

Specifies the transforms to apply to the element. It is a space-separated list of transforms, which are applied one after the other, as requested by the composition operation. Composite transforms are effectively applied in order from right to left.

Examples

Here is an unaltered image of the MDN logo:



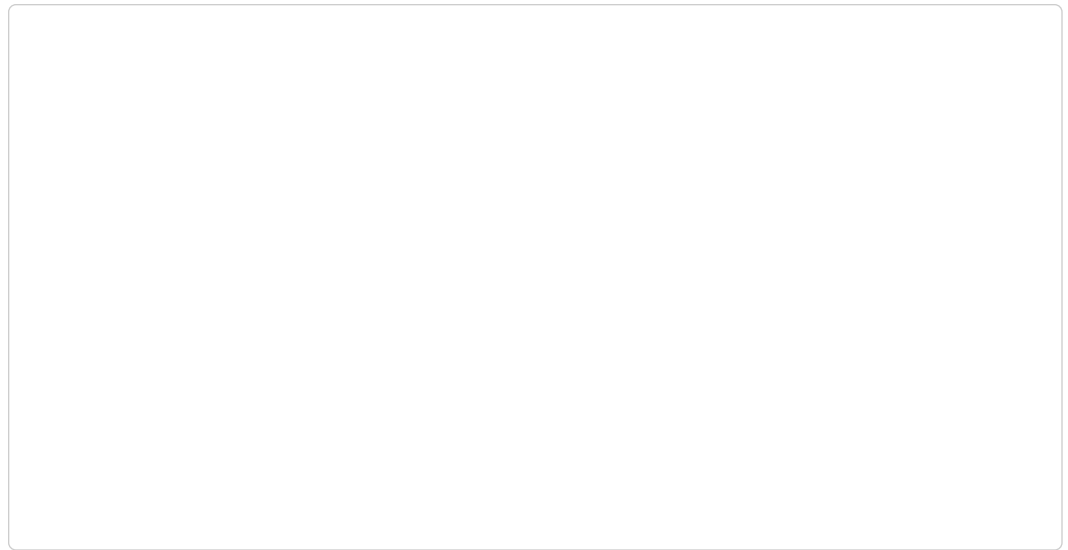
Rotating

Here is the MDN logo rotated 90 degrees from its bottom-left corner.

```

```



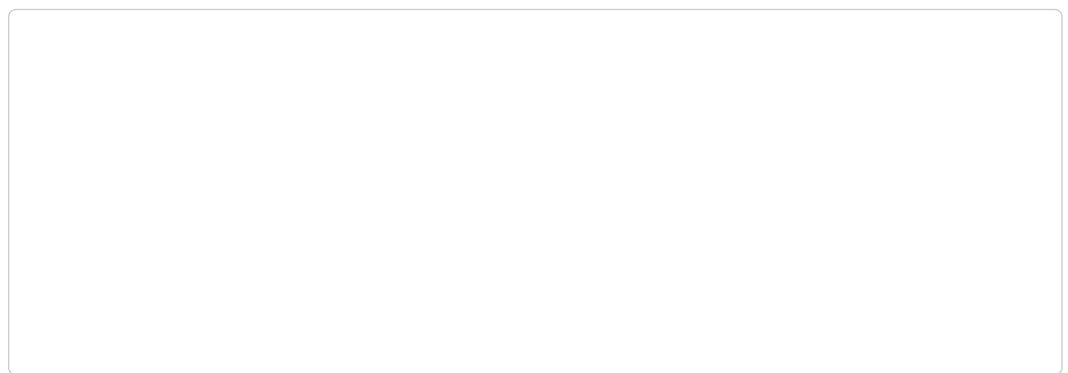


Skewing and translating

Here is the MDN logo, skewed by 10 degrees and translated by 150 pixels on the X-axis.

```

```



3D specific CSS properties

Performing CSS transformations in 3D space is a bit more complex. You have to start by configuring the 3D space by giving it a perspective, then you have to configure how your 2D elements will

behave in that space.

Perspective

The first element to set is the [perspective](#). The perspective is what gives us the 3D impression. The farther from the viewer the elements are, the smaller they are.

Setting perspective

This example shows a cube with the perspective set at different positions. How quick the cube shrinks is defined by the [perspective](#) property. The smaller its value is, the deeper the perspective is.

HTML

The HTML below creates four copies of the same box, with the perspective set at different values.

```
<table>
  <tbody>
    <tr>
      <th><code>perspective: 250px;</code></th>
      <th><code>perspective: 350px;</code></th>
    </tr>
    <tr>
      <td>
        <div class="container">
          <div class="cube pers250">
            <div class="face front">1</div>
            <div class="face back">2</div>
            <div class="face right">3</div>
            <div class="face left">4</div>
            <div class="face top">5</div>
            <div class="face bottom">6</div>
          </div>
        </div>
      </td>
    </tr>
  </tbody>
</table>
```

```
<td>
  <div class="container">
    <div class="cube pers350">
      <div class="face front">1</div>
      <div class="face back">2</div>
      <div class="face right">3</div>
      <div class="face left">4</div>
      <div class="face top">5</div>
      <div class="face bottom">6</div>
    </div>
  </div>
</td>
</tr>
<tr>
  <th><code>perspective: 500px;</code></th>
  <th><code>perspective: 650px;</code></th>
</tr>
<tr>
  <td>
    <div class="container">
      <div class="cube pers500">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
        <div class="face top">5</div>
        <div class="face bottom">6</div>
      </div>
    </div>
  </td>
  <td>
    <div class="container">
      <div class="cube pers650">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
        <div class="face top">5</div>
        <div class="face bottom">6</div>
      </div>
    </div>
  </td>
</tr>
```

```
    </td>
  </tr>
</tbody>
</table>
```

CSS

The CSS establishes classes that can be used to set the perspective to different distances. It also includes classes for the container box and the cube itself, as well as each of its faces.

```
/* Shorthand classes for different perspective values */
.pers250 {
  perspective: 250px;
}

.pers350 {
  perspective: 350px;
}

.pers500 {
  perspective: 500px;
}

.pers650 {
  perspective: 650px;
}

/* Define the container div, the cube div, and a generic face */
.container {
  width: 200px;
  height: 200px;
  margin: 75px 0 0 75px;
  border: none;
}

.cube {
  width: 100%;
  height: 100%;
```

```
    backface-visibility: visible;
    perspective-origin: 150% 150%;
    transform-style: preserve-3d;
}

.face {
    display: block;
    position: absolute;
    width: 100px;
    height: 100px;
    border: none;
    line-height: 100px;
    font-family: sans-serif;
    font-size: 60px;
    color: white;
    text-align: center;
}

/* Define each face based on direction */
.front {
    background: rgba(0, 0, 0, 0.3);
    transform: translateZ(50px);
}

.back {
    background: rgba(0, 255, 0, 1);
    color: black;
    transform: rotateY(180deg) translateZ(50px);
}

.right {
    background: rgba(196, 0, 0, 0.7);
    transform: rotateY(90deg) translateZ(50px);
}

.left {
    background: rgba(0, 0, 196, 0.7);
    transform: rotateY(-90deg) translateZ(50px);
}

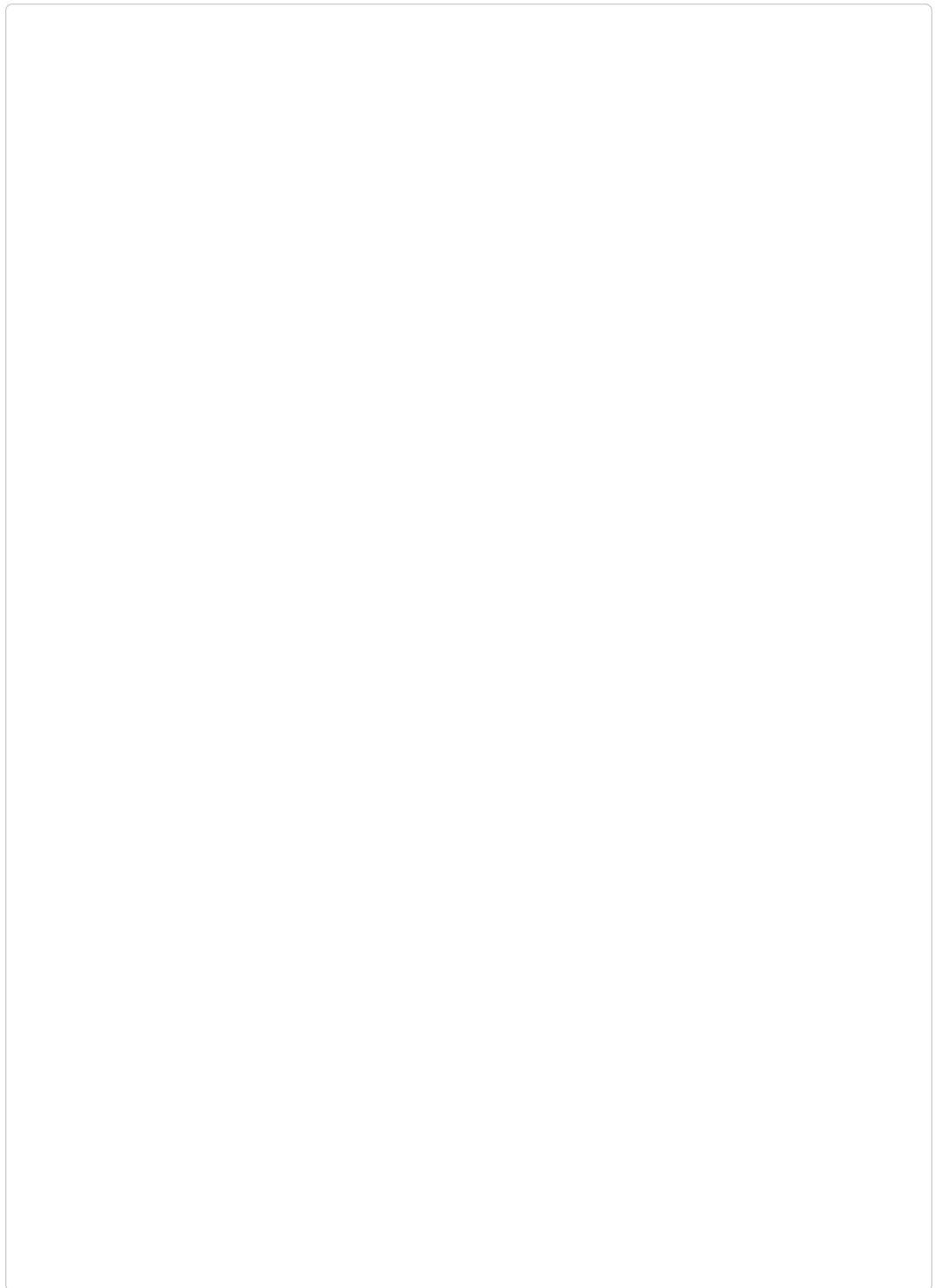
.top {
```

```
    background: rgba(196, 196, 0, 0.7);
    transform: rotateX(90deg) translateZ(50px);
}

.bottom {
    background: rgba(196, 0, 196, 0.7);
    transform: rotateX(-90deg) translateZ(50px);
}

/* Make the table a little nicer */
th,
p,
td {
    background-color: #eeeeee;
    padding: 10px;
    font-family: sans-serif;
    text-align: left;
}
```

RESULT



The second element to configure is the position of the viewer, with the `perspective-origin` property. By default the perspective is centered on the viewer, which is not always adequate.

Changing the perspective origin

This example shows cubes with popular `perspective-origin` values.

HTML

```
<section>
  <figure>
    <figcaption><code>perspective-origin: top left;</code></figcaption>
    <div class="container">
      <div class="cube potl">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
        <div class="face top">5</div>
        <div class="face bottom">6</div>
      </div>
    </div>
  </figure>

  <figure>
    <figcaption><code>perspective-origin: top;</code></figcaption>
    <div class="container">
      <div class="cube potm">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
        <div class="face top">5</div>
        <div class="face bottom">6</div>
      </div>
    </div>
  </figure>

  <figure>
    <figcaption><code>perspective-origin: top right;</code>
  </figcaption>
    <div class="container">
      <div class="cube potr">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
        <div class="face top">5</div>
```

```
        <div class="face bottom">6</div>
      </div>
    </div>
  </figure>

  <figure>
    <figcaption><code>perspective-origin: left;</code></figcaption>
    <div class="container">
      <div class="cube poml">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
        <div class="face top">5</div>
        <div class="face bottom">6</div>
      </div>
    </div>
  </figure>

  <figure>
    <figcaption><code>perspective-origin: 50% 50%;</code></figcaption>
    <div class="container">
      <div class="cube pomm">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
        <div class="face top">5</div>
        <div class="face bottom">6</div>
      </div>
    </div>
  </figure>

  <figure>
    <figcaption><code>perspective-origin: right;</code></figcaption>
    <div class="container">
      <div class="cube pomr">
        <div class="face front">1</div>
        <div class="face back">2</div>
        <div class="face right">3</div>
        <div class="face left">4</div>
```

```
<div class="face top">5</div>
<div class="face bottom">6</div>
</div>
</div>
</figure>

<figure>
  <figcaption><code>perspective-origin: bottom left;</code>
</figcaption>
  <div class="container">
    <div class="cube pobl">
      <div class="face front">1</div>
      <div class="face back">2</div>
      <div class="face right">3</div>
      <div class="face left">4</div>
      <div class="face top">5</div>
      <div class="face bottom">6</div>
    </div>
  </div>
</figure>

<figure>
  <figcaption><code>perspective-origin: bottom;</code></figcaption>
  <div class="container">
    <div class="cube pobm">
      <div class="face front">1</div>
      <div class="face back">2</div>
      <div class="face right">3</div>
      <div class="face left">4</div>
      <div class="face top">5</div>
      <div class="face bottom">6</div>
    </div>
  </div>
</figure>

<figure>
  <figcaption><code>perspective-origin: bottom right;</code>
</figcaption>
  <div class="container">
    <div class="cube pobr">
      <div class="face front">1</div>
```

```
<div class="face back">2</div>
<div class="face right">3</div>
<div class="face left">4</div>
<div class="face top">5</div>
<div class="face bottom">6</div>
</div>
</div>
</figure>

<figure>
  <figcaption><code>perspective-origin: -200% -200%;</code>
</figcaption>
  <div class="container">
    <div class="cube po200200neg">
      <div class="face front">1</div>
      <div class="face back">2</div>
      <div class="face right">3</div>
      <div class="face left">4</div>
      <div class="face top">5</div>
      <div class="face bottom">6</div>
    </div>
  </div>
</figure>

<figure>
  <figcaption><code>perspective-origin: 200% 200%;</code>
</figcaption>
  <div class="container">
    <div class="cube po200200pos">
      <div class="face front">1</div>
      <div class="face back">2</div>
      <div class="face right">3</div>
      <div class="face left">4</div>
      <div class="face top">5</div>
      <div class="face bottom">6</div>
    </div>
  </div>
</figure>

<figure>
  <figcaption><code>perspective-origin: 200% -200%;</code>
```

```
</figcaption>
<div class="container">
  <div class="cube po200200">
    <div class="face front">1</div>
    <div class="face back">2</div>
    <div class="face right">3</div>
    <div class="face left">4</div>
    <div class="face top">5</div>
    <div class="face bottom">6</div>
  </div>
</div>
</figure>
</section>
```

CSS

```
/* perspective-origin values (unique per example) */
.potl {
  perspective-origin: top left;
}
.potm {
  perspective-origin: top;
}
.potr {
  perspective-origin: top right;
}
.poml {
  perspective-origin: left;
}
.pomm {
  perspective-origin: 50% 50%;
}
.pomr {
  perspective-origin: right;
}
.pobl {
  perspective-origin: bottom left;
}
.pobm {
  perspective-origin: bottom;
```

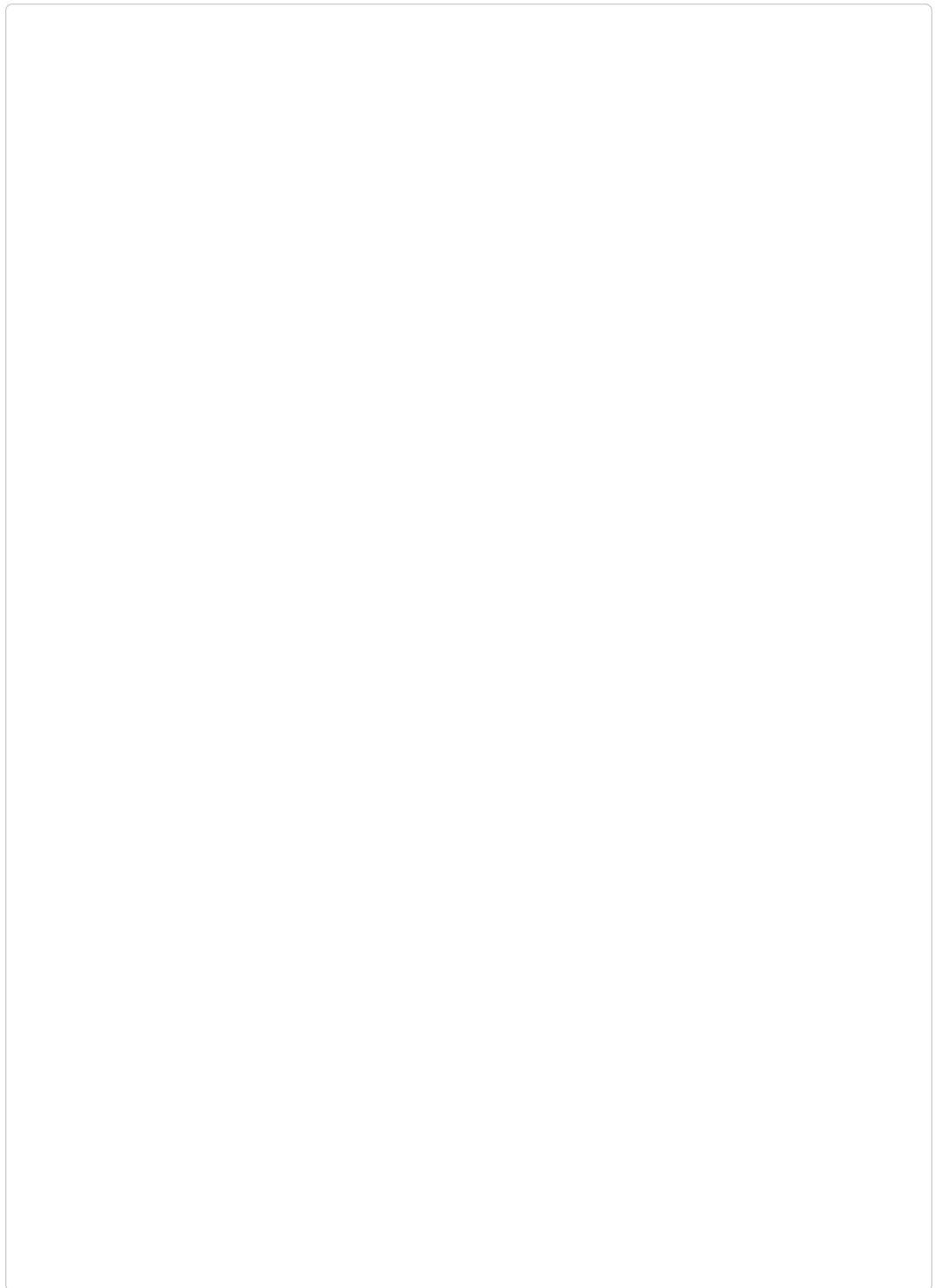
```
}  
.pobr {  
  perspective-origin: bottom right;  
}  
.po200200neg {  
  perspective-origin: -200% -200%;  
}  
.po200200pos {  
  perspective-origin: 200% 200%;  
}  
.po200200 {  
  perspective-origin: 200% -200%;  
}  
  
/* Define the container div, the cube div, and a generic face */  
.container {  
  width: 100px;  
  height: 100px;  
  margin: 24px;  
  border: none;  
}  
  
.cube {  
  width: 100%;  
  height: 100%;  
  backface-visibility: visible;  
  perspective: 300px;  
  transform-style: preserve-3d;  
}  
  
.face {  
  display: block;  
  position: absolute;  
  width: 100px;  
  height: 100px;  
  border: none;  
  line-height: 100px;  
  font-family: sans-serif;  
  font-size: 60px;  
  color: white;  
  text-align: center;
```

```
}

/* Define each face based on direction */
.front {
  background: rgba(0, 0, 0, 0.3);
  transform: translateZ(50px);
}
.back {
  background: rgba(0, 255, 0, 1);
  color: black;
  transform: rotateY(180deg) translateZ(50px);
}
.right {
  background: rgba(196, 0, 0, 0.7);
  transform: rotateY(90deg) translateZ(50px);
}
.left {
  background: rgba(0, 0, 196, 0.7);
  transform: rotateY(-90deg) translateZ(50px);
}
.top {
  background: rgba(196, 196, 0, 0.7);
  transform: rotateX(90deg) translateZ(50px);
}
.bottom {
  background: rgba(196, 0, 196, 0.7);
  transform: rotateX(-90deg) translateZ(50px);
}

/* Make the layout a little nicer */
section {
  background-color: #eee;
  padding: 10px;
  font-family: sans-serif;
  text-align: left;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```



RESULT



Once you have done this, you can work on the element in the 3D space.

See also

- The [CSS `transform` property](#) and the [CSS `<transform-function>` data types](#)

- The individual transforms properties: [translate](#), [rotate](#), and [scale](#) (There is no `skew` property)
- [Using device orientation with 3D Transforms](#)
- [Intro to CSS 3D transforms](#)  (Blog post by David DeSandro)
- [CSS Transform Playground](#)  (Online tool to visualize CSS Transform functions)

This page was last modified on Feb 21, 2023 by [MDN contributors](#).