Module 02

Purwadhika
Digital Technology School

Full Stack Web Development

Advanced CSS

Job Connector Program

Outline

- Media Queries
- Grid & Flexbox
- Animation
- Gradient

Media Queries

Media queries are useful when you want to modify your site or app depending on a device's general type (such as print vs. screen) or specific characteristics and parameters (such as screen resolution or browser viewport width).

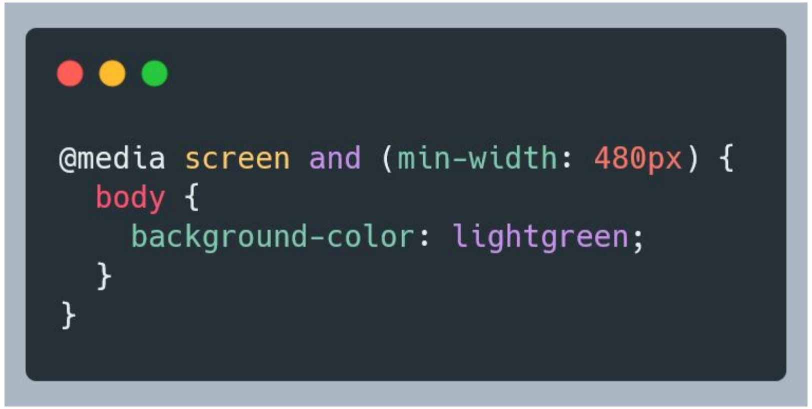
Media queries are used for the following:

- To conditionally apply styles with the CSS @media and @import at-rules.
- To target specific media for the <style>, <link>, <source>, and other HTML elements with the media= attribute.
- To test and monitor media states using the Window.matchMedia() and MediaQueryList.addListener() JavaScript methods.

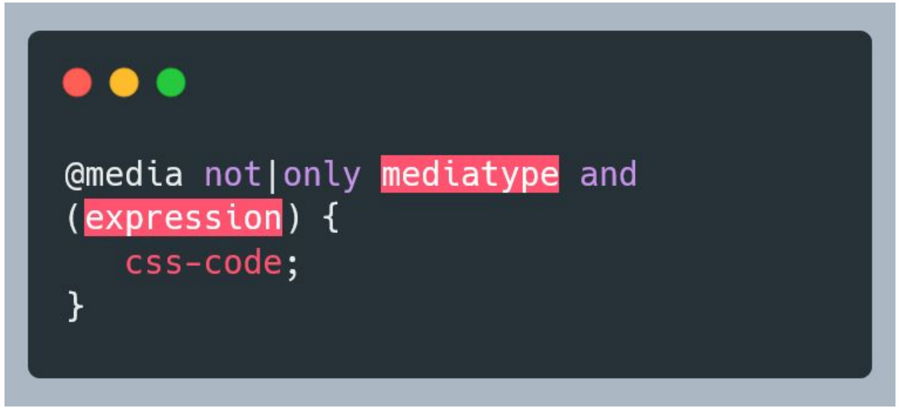
Src: https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

Media Query Syntax

Example:



```
@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
```



```
@media not|only mediatype and
(expression) {
    css-code;
}
```

The following example shows a menu that will float to the left of the page if the **viewport** is 480 pixels wide or wider (if the **viewport** is less than 480 pixels, the menu will be on top of the content):

Common Media Query Breakpoints

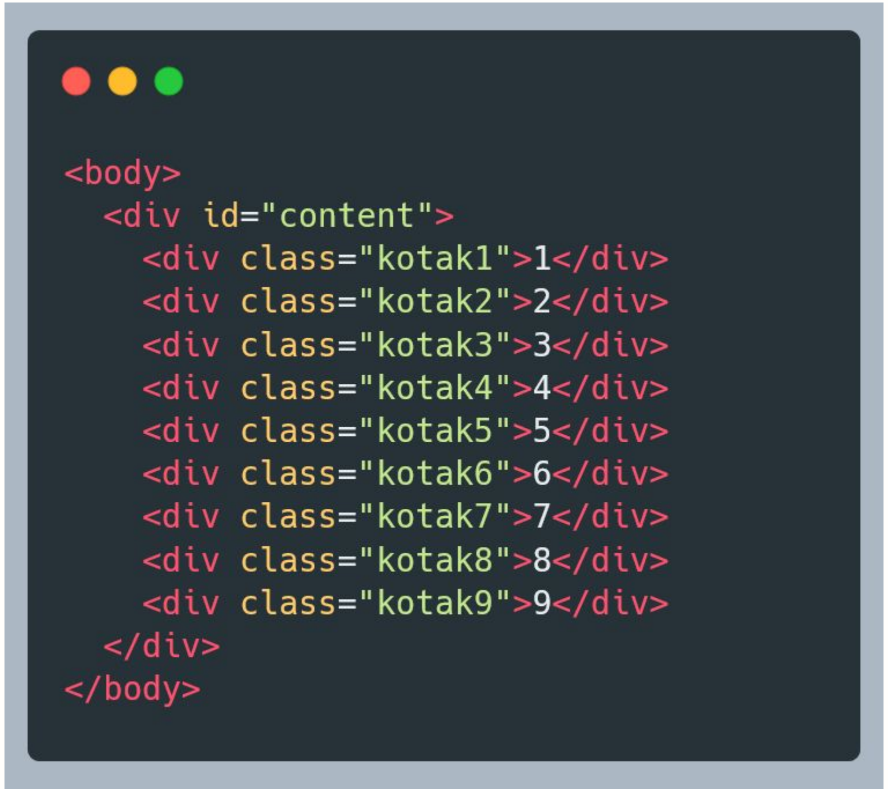
There is no standard exactly defined, but these are some **commonly used ones**

- 320px—480px: Mobile devices
- 481px—768px: iPads, Tablets
- 769px—1024px: Small screens, laptops
- 1025px—1200px: Desktops, large screens
- 1201px and more— Extra large screens, TV

Grid

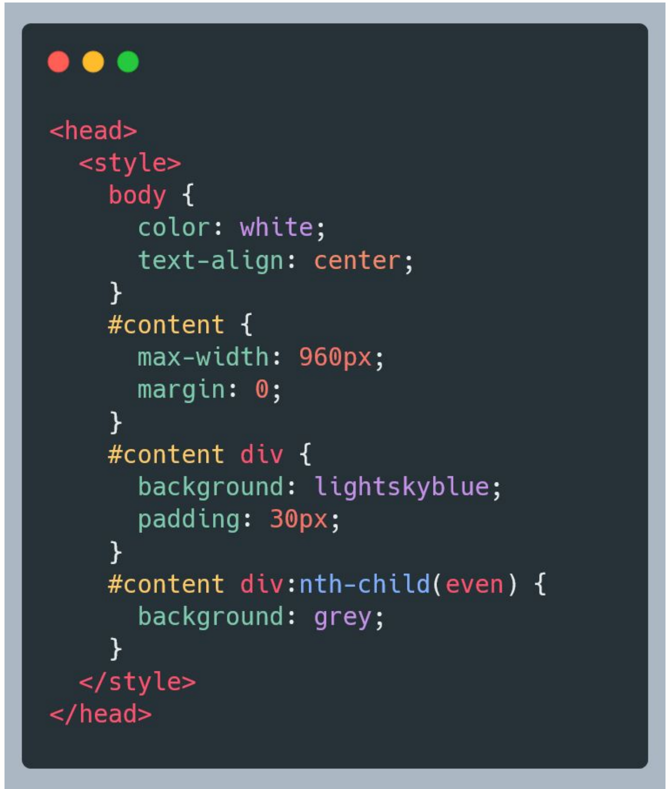
CSS Grid Layout (aka “Grid” or “CSS Grid”), is a two-dimensional grid-based layout system that, compared to any web layout system of the past, completely changes the way we design user interfaces.

Grid Example



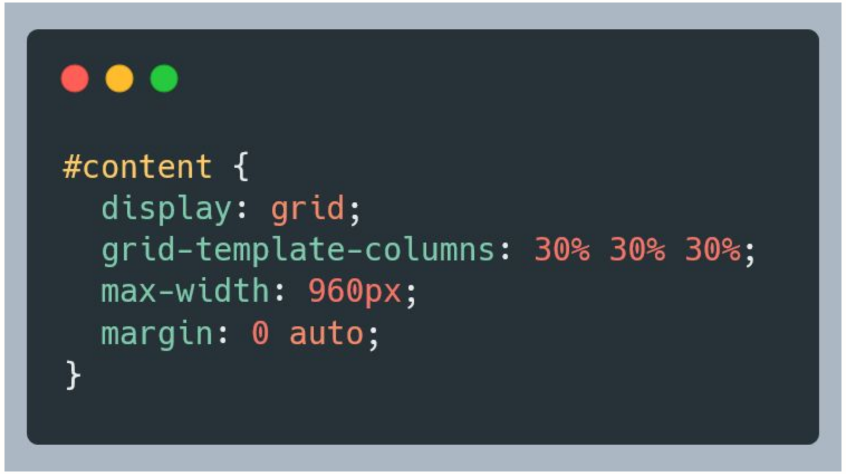
```
<body>
  <div id="content">
    <div class="kotak1">1</div>
    <div class="kotak2">2</div>
    <div class="kotak3">3</div>
    <div class="kotak4">4</div>
    <div class="kotak5">5</div>
    <div class="kotak6">6</div>
    <div class="kotak7">7</div>
    <div class="kotak8">8</div>
    <div class="kotak9">9</div>
  </div>
</body>
```

Grid Example



```
<head>
  <style>
    body {
      color: white;
      text-align: center;
    }
    #content {
      max-width: 960px;
      margin: 0;
    }
    #content div {
      background: lightskyblue;
      padding: 30px;
    }
    #content div:nth-child(even) {
      background: grey;
    }
  </style>
</head>
```

Grid Template Columns



```
#content {
  display: grid;
  grid-template-columns: 30% 30% 30%;
  max-width: 960px;
  margin: 0 auto;
}
```

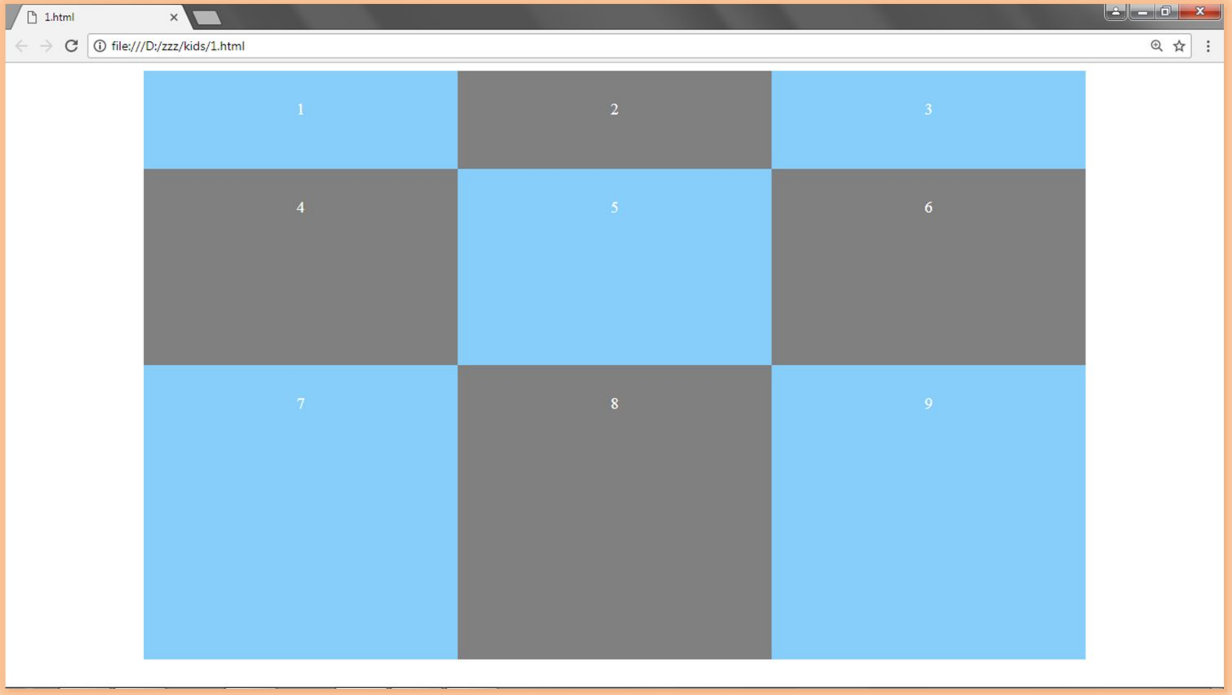
= = = = = *Using Fraction:* = = = = =

```
grid-template-columns: 1fr 2fr 1fr;
```

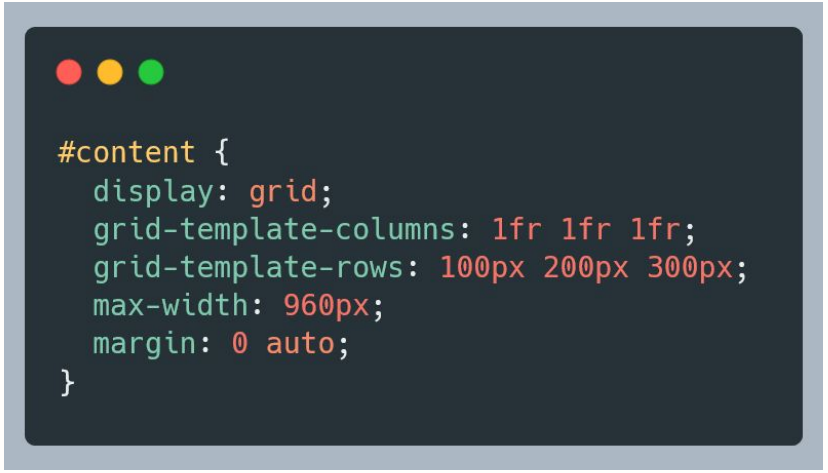
= = = = = *Using Repeat:* = = = = =

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

Grid Template Rows



Grid Template Rows



```css
#content {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100px 200px 300px;
  max-width: 960px;
  margin: 0 auto;
}
```

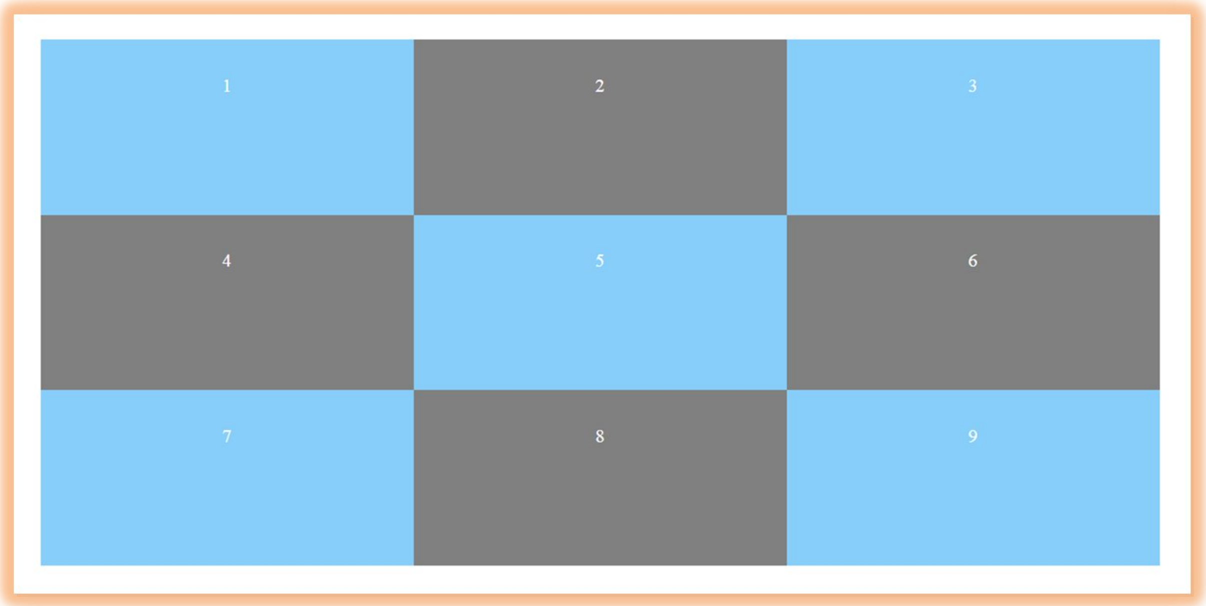
= = = = = *Using Fraction:* = = = = =

```css
grid-template-rows: 1fr 2fr 1fr;
```

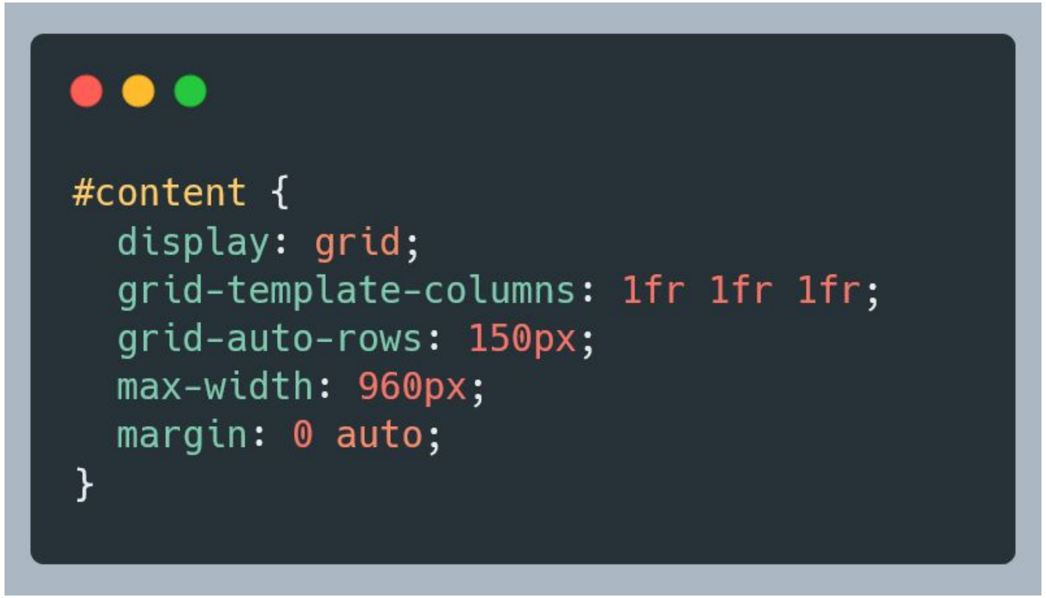
= = = = = *Using Repeat:* = = = = =

```css
grid-template-rows: repeat(3, 1fr);
```

Grid Auto Rows



Grid Auto Rows

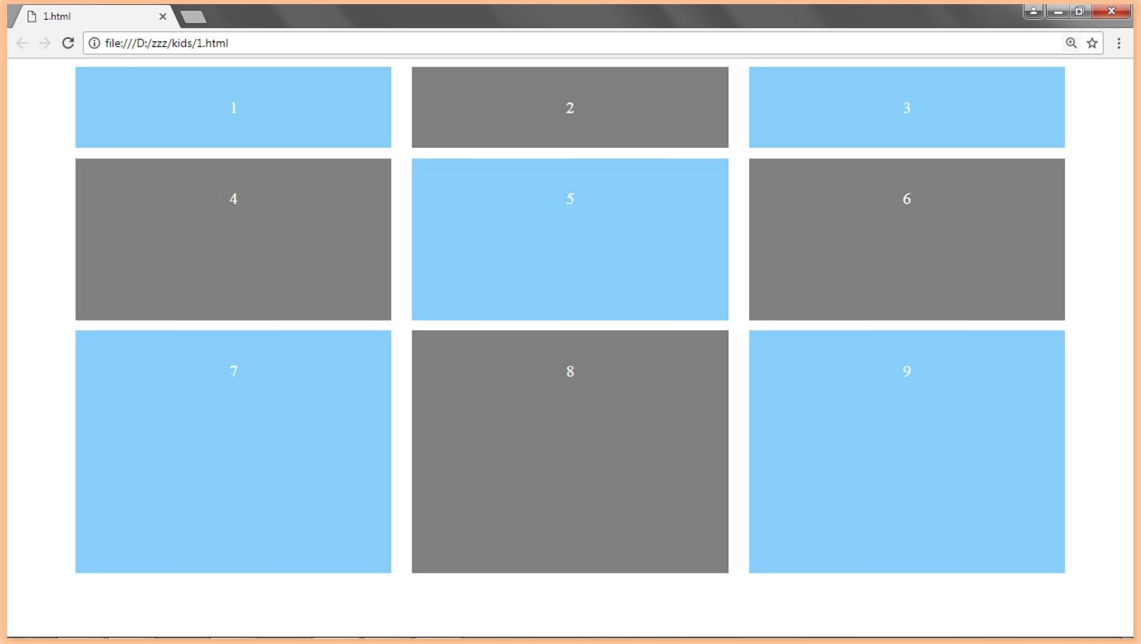


```
#content {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-auto-rows: 150px;
  max-width: 960px;
  margin: 0 auto;
}
```

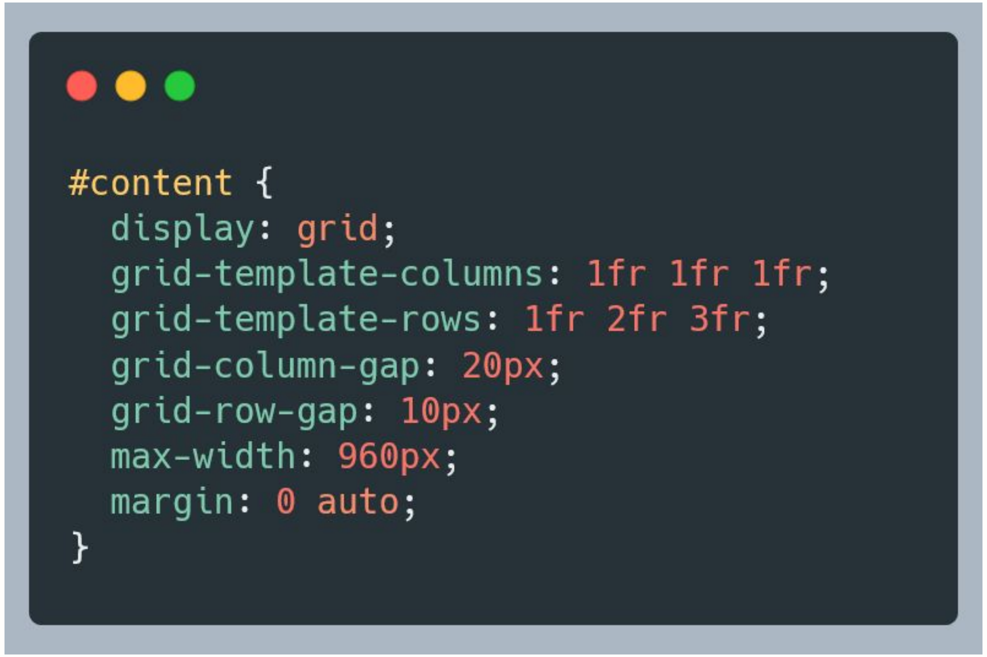
= = = = = *Using MinMax:* = = = = =

```
grid-auto-rows: minmax(100px, auto);
```

Grid Gap



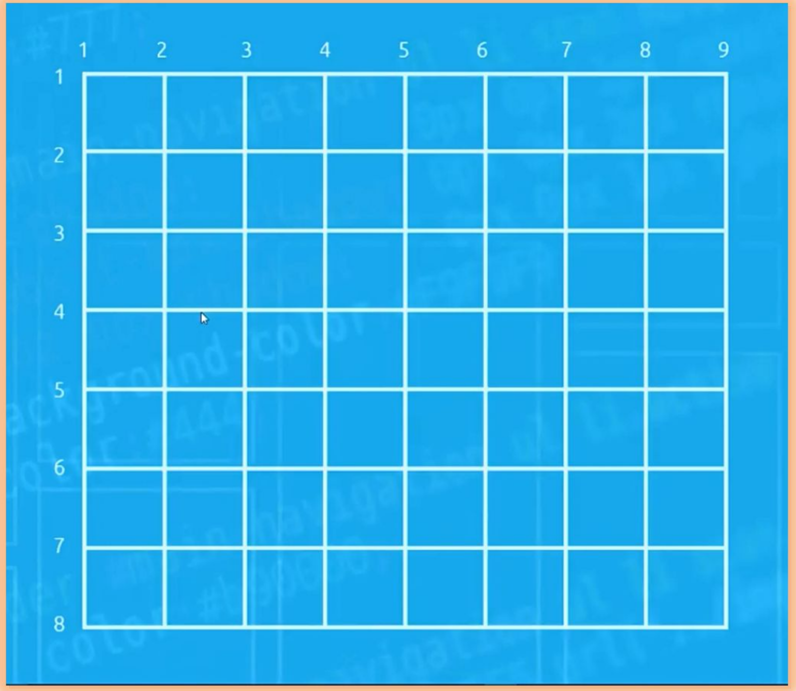
Grid Gap



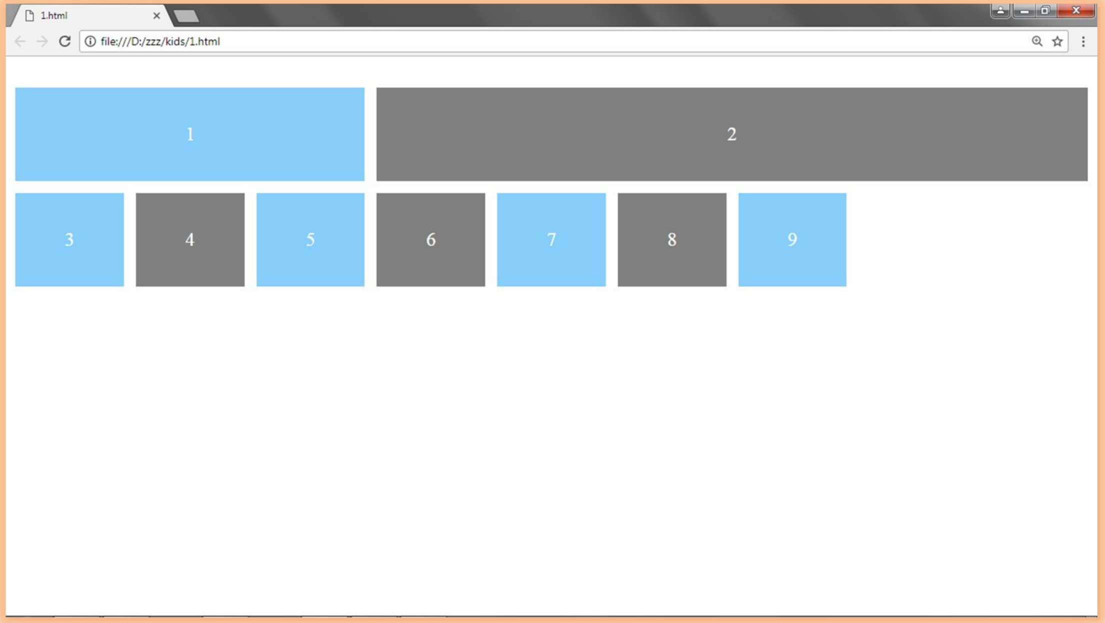
```
#content {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 1fr 2fr 3fr;
  grid-column-gap: 20px;
  grid-row-gap: 10px;
  max-width: 960px;
  margin: 0 auto;
}
```

```
// grid-gap: 20px;
```

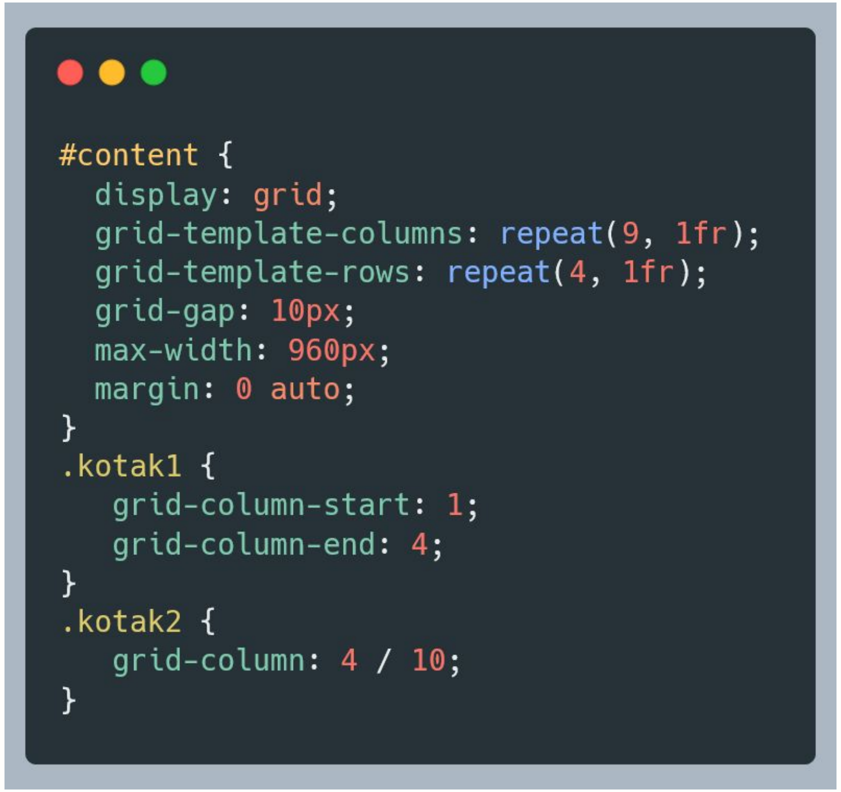
Grid Lines



Grid Column

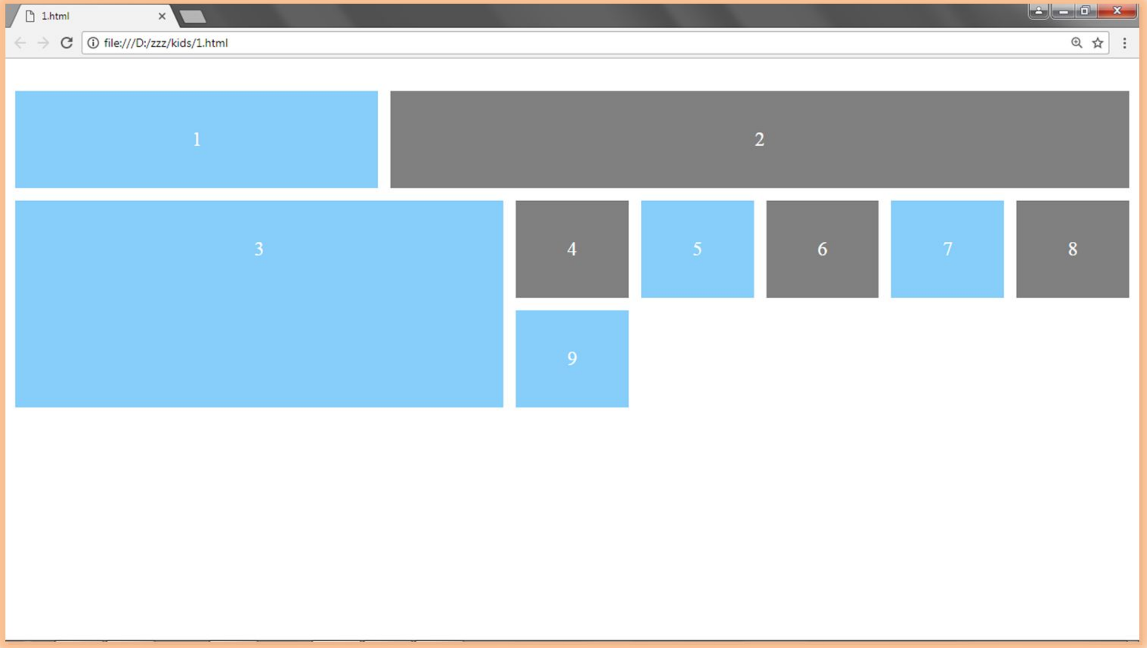


Grid Column

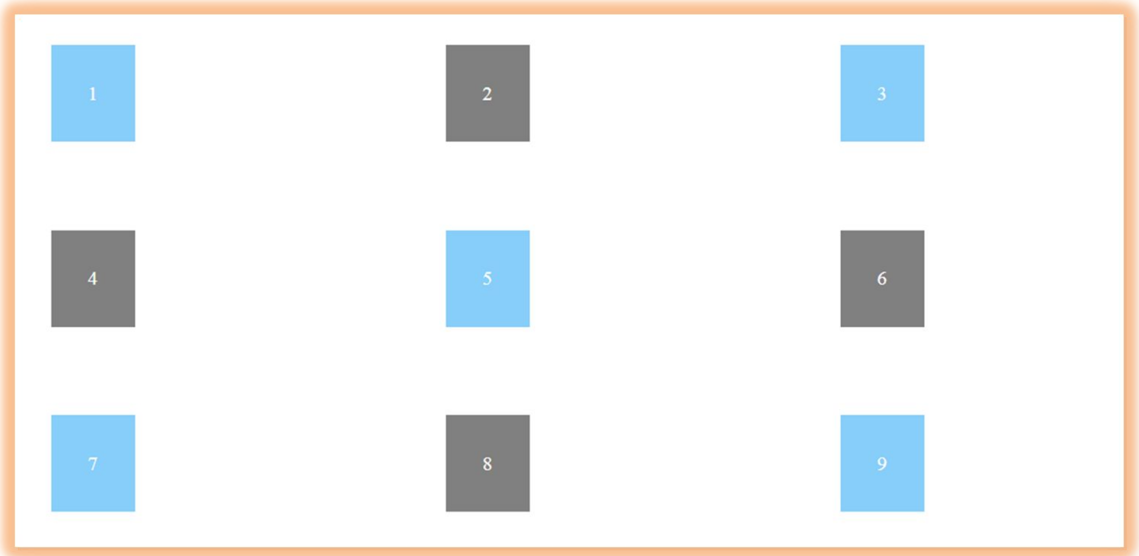


```
#content {
  display: grid;
  grid-template-columns: repeat(9, 1fr);
  grid-template-rows: repeat(4, 1fr);
  grid-gap: 10px;
  max-width: 960px;
  margin: 0 auto;
}
.kotak1 {
   grid-column-start: 1;
   grid-column-end: 4;
}
.kotak2 {
   grid-column: 4 / 10;
}
```

Grid Row

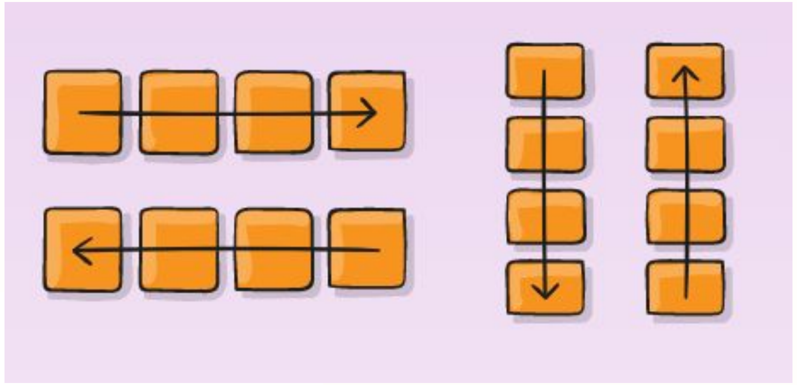


Grid Justify Items



Flex Directions

This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

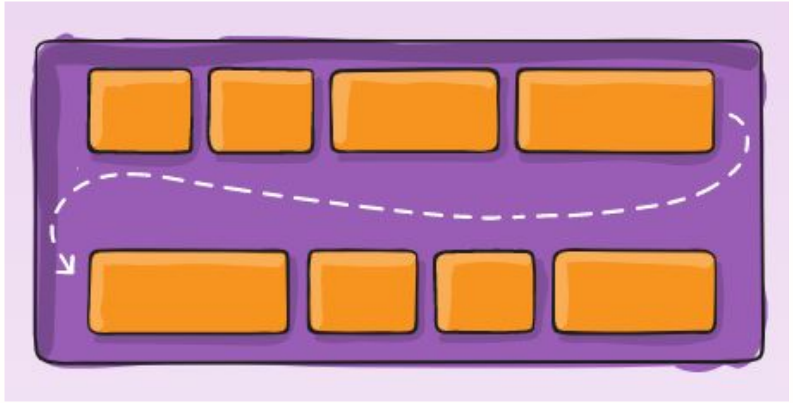


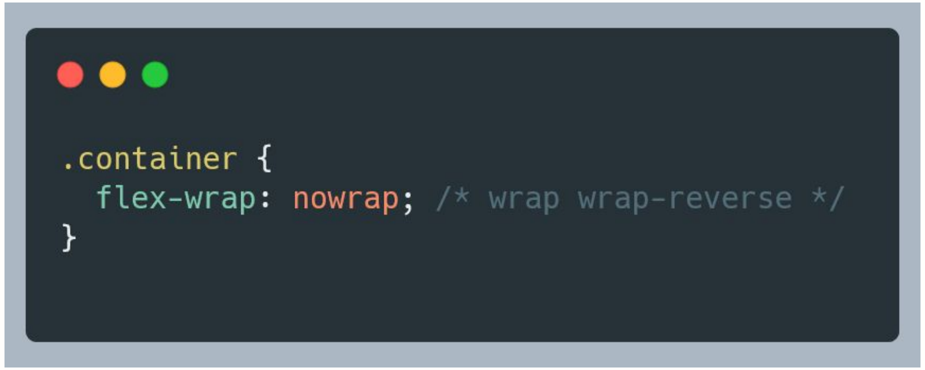
CSS Syntax

```
flex-direction: row|row-reverse|column|column-reverse|initial|inherit;
```

Flex-wrap

By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

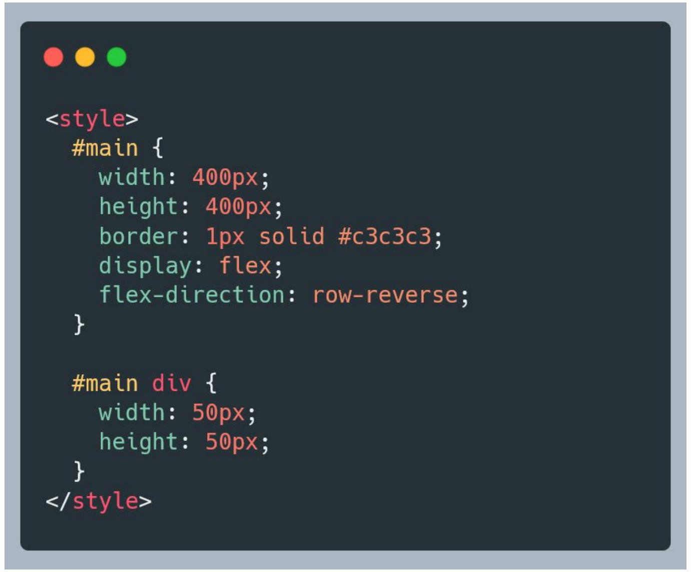




```
.container {
  flex-wrap: nowrap; /* wrap wrap-reverse */
}
```

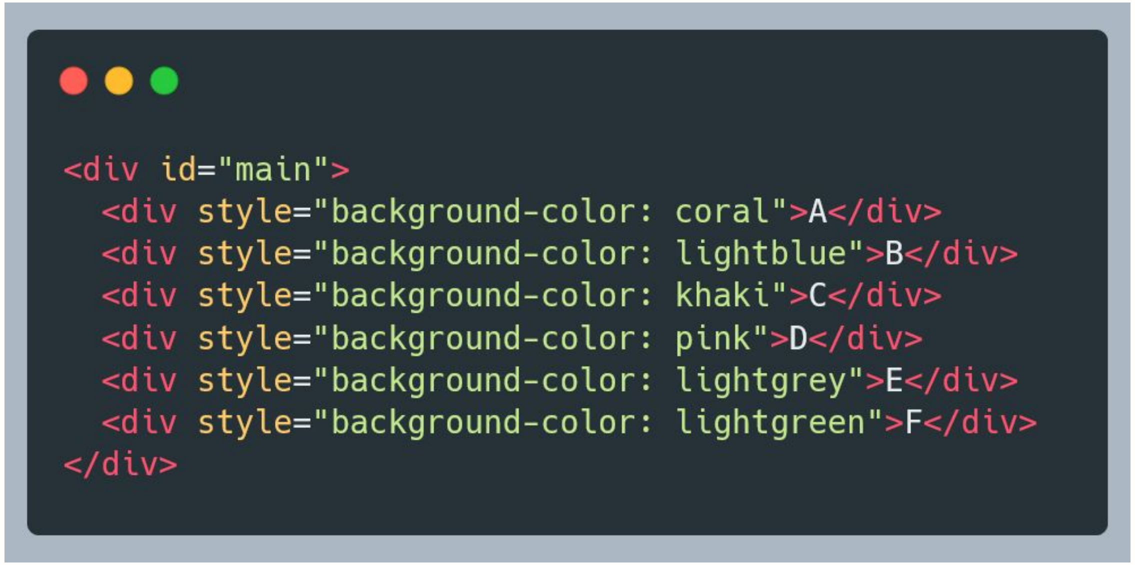
Find out more on: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flex Example



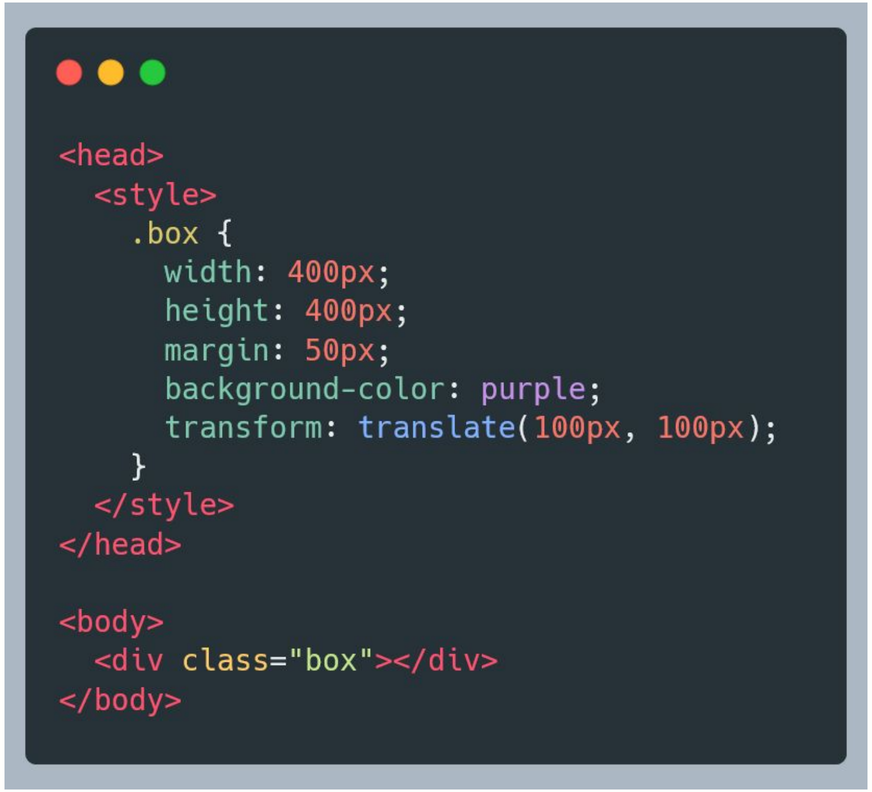
```
<style>
  #main {
    width: 400px;
    height: 400px;
    border: 1px solid #c3c3c3;
    display: flex;
    flex-direction: row-reverse;
  }

  #main div {
    width: 50px;
    height: 50px;
  }
</style>
```



```
<div id="main">
  <div style="background-color: coral">A</div>
  <div style="background-color: lightblue">B</div>
  <div style="background-color: khaki">C</div>
  <div style="background-color: pink">D</div>
  <div style="background-color: lightgrey">E</div>
  <div style="background-color: lightgreen">F</div>
</div>
```

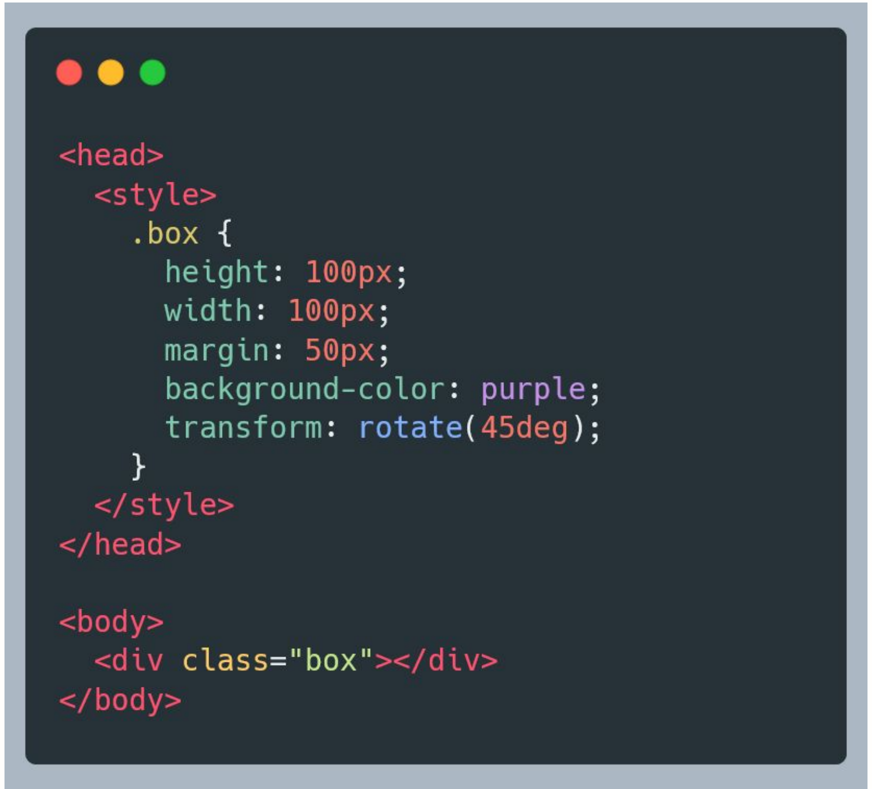
Animation – 2D Transform – Translate



```
<head>
  <style>
    .box {
      width: 400px;
      height: 400px;
      margin: 50px;
      background-color: purple;
      transform: translate(100px, 100px);
    }
  </style>
</head>

<body>
  <div class="box"></div>
</body>
```

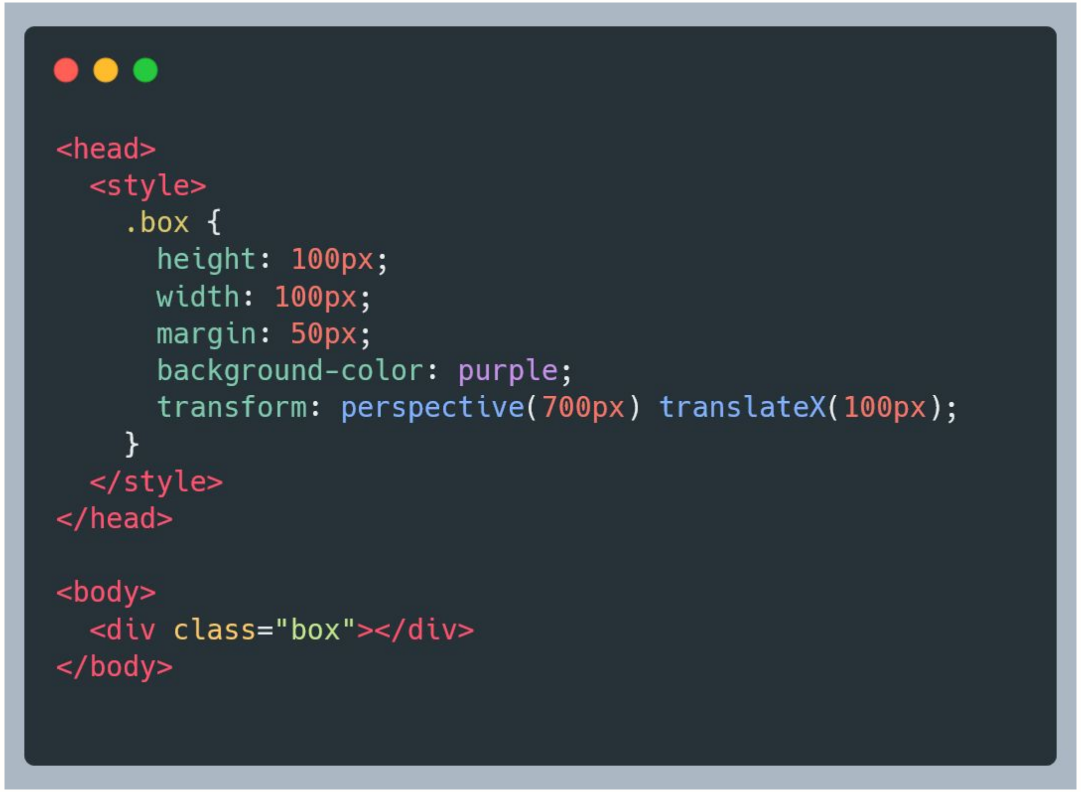
Animation – 2D Transform – Rotate



```
<head>
  <style>
    .box {
      height: 100px;
      width: 100px;
      margin: 50px;
      background-color: purple;
      transform: rotate(45deg);
    }
  </style>
</head>

<body>
  <div class="box"></div>
</body>
```

Animation – 3D Transforms – TranslateX



```
<head>
  <style>
    .box {
      height: 100px;
      width: 100px;
      margin: 50px;
      background-color: purple;
      transform: perspective(700px) translateX(100px);
    }
  </style>
</head>

<body>
  <div class="box"></div>
</body>
```