**CSS**

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

**Tip:** The word **cascading** means that a style applied to a parent element will also apply to all children elements within the parent. So, if you set the color of the body text to "blue", all headings, paragraphs, and other text elements within the body will also get the same color (unless you specify something else)!

Using CSS

CSS can be added to HTML documents in 3 ways:

* **Inline** - by using the style attribute inside HTML elements
* **Internal** - by using a <style> element in the <head> section
* **External** - by using a <link> element to link to an external CSS file



**Inline**

<body>

  <h1 style="color: blue;">Style Me in Blue!</h1>

</body>

**Internal**

<head>

  <meta charset="UTF-8">

  <title>Internal</title>

  <style>

    h1 {

      color: red;

    }

  </style>

</head>

<body>

  <h1>Style Me in Red!</h1>

</body>

**External**

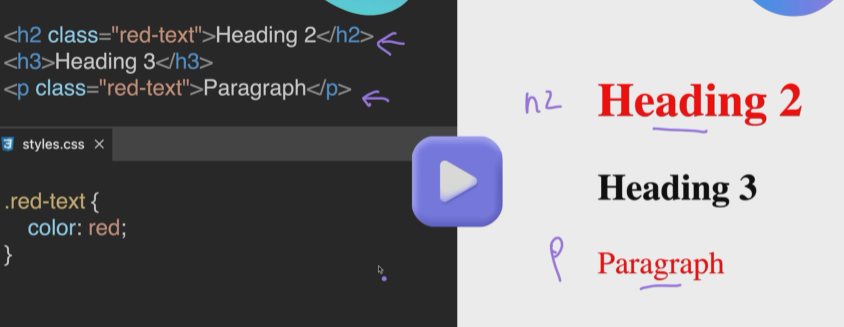
<head>

  <title>External</title>

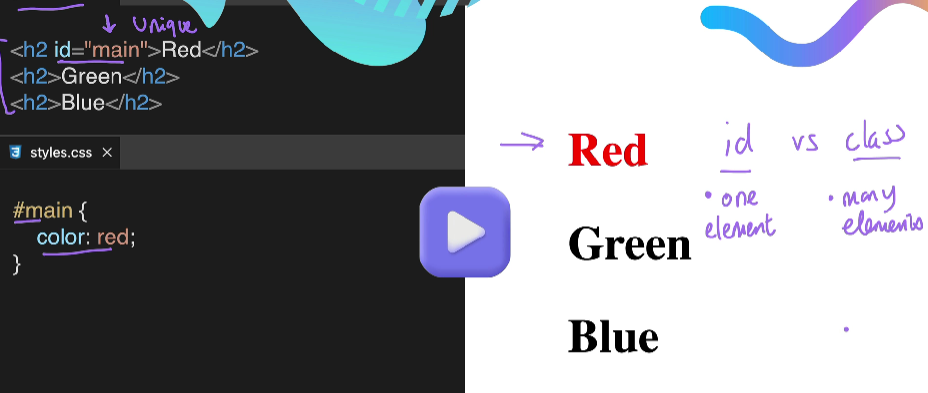
  <link rel="stylesheet" href="./style.css" />

</head>

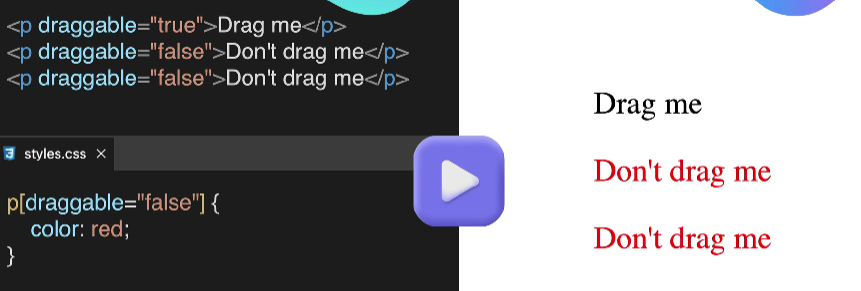
**CSS Selectors**



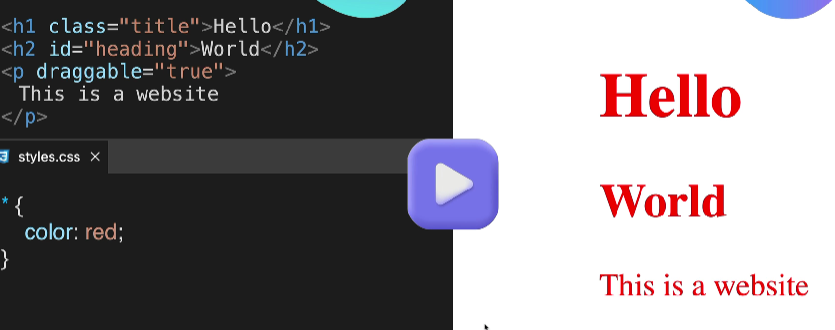
**Id Selector**

****

**Attribute Selector**



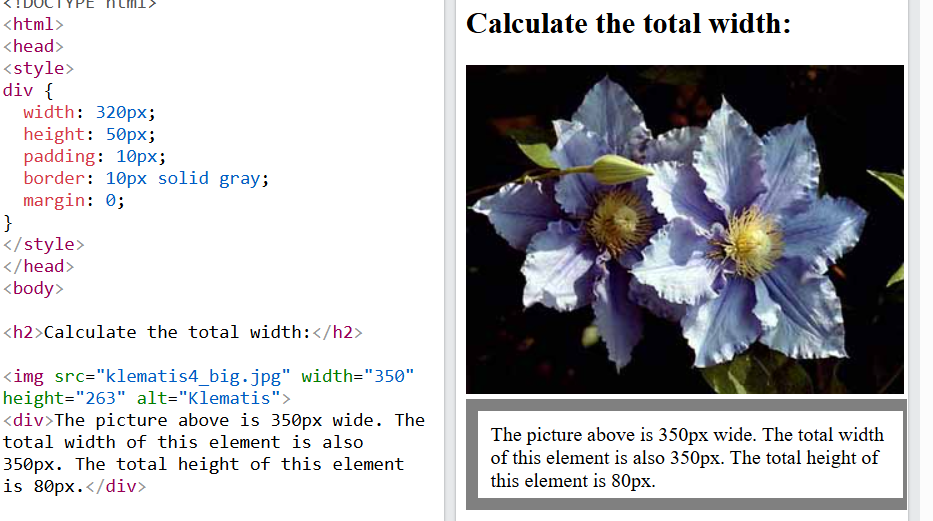
**Universal Selector**



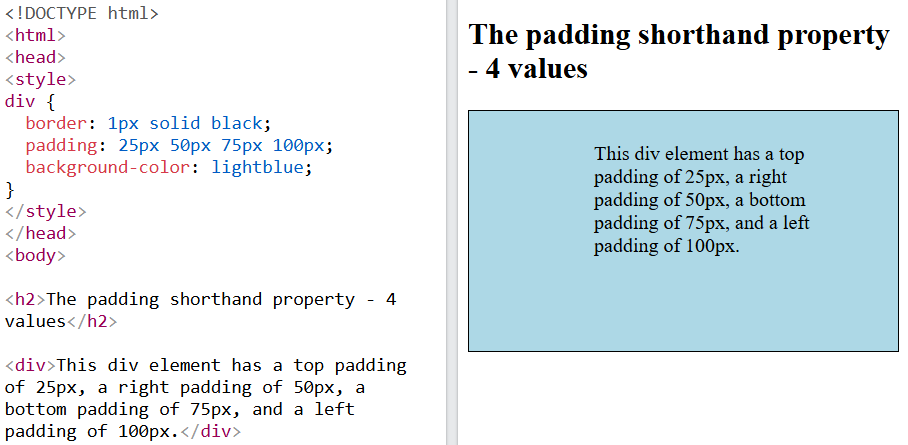


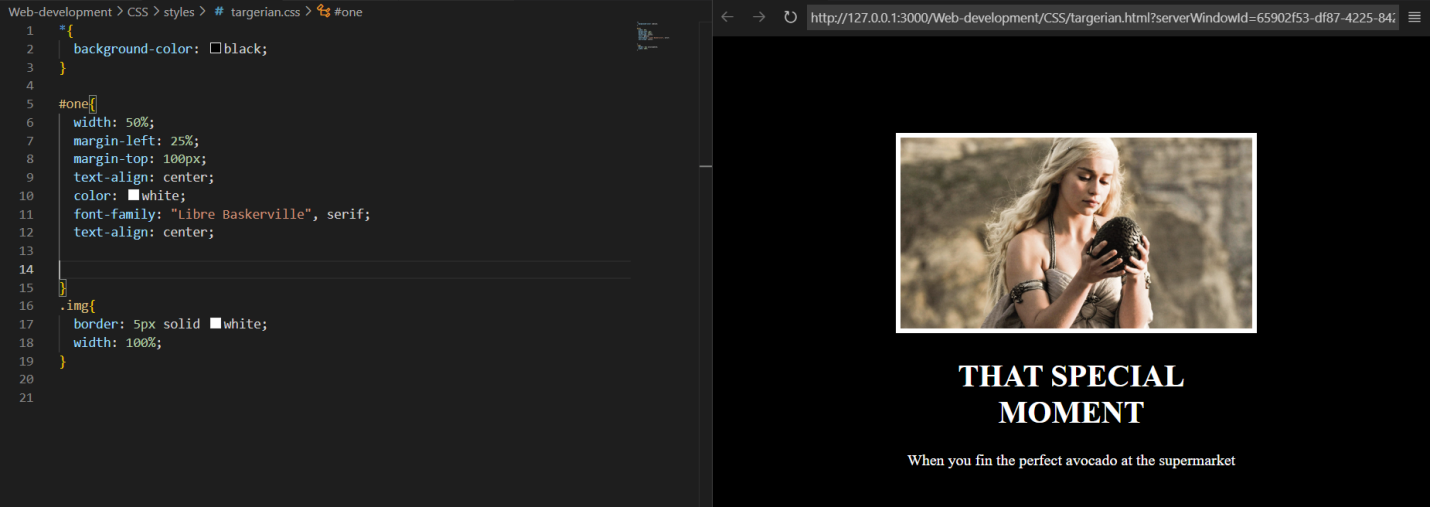


**The CSS Box Model - Margin, Padding and Border**

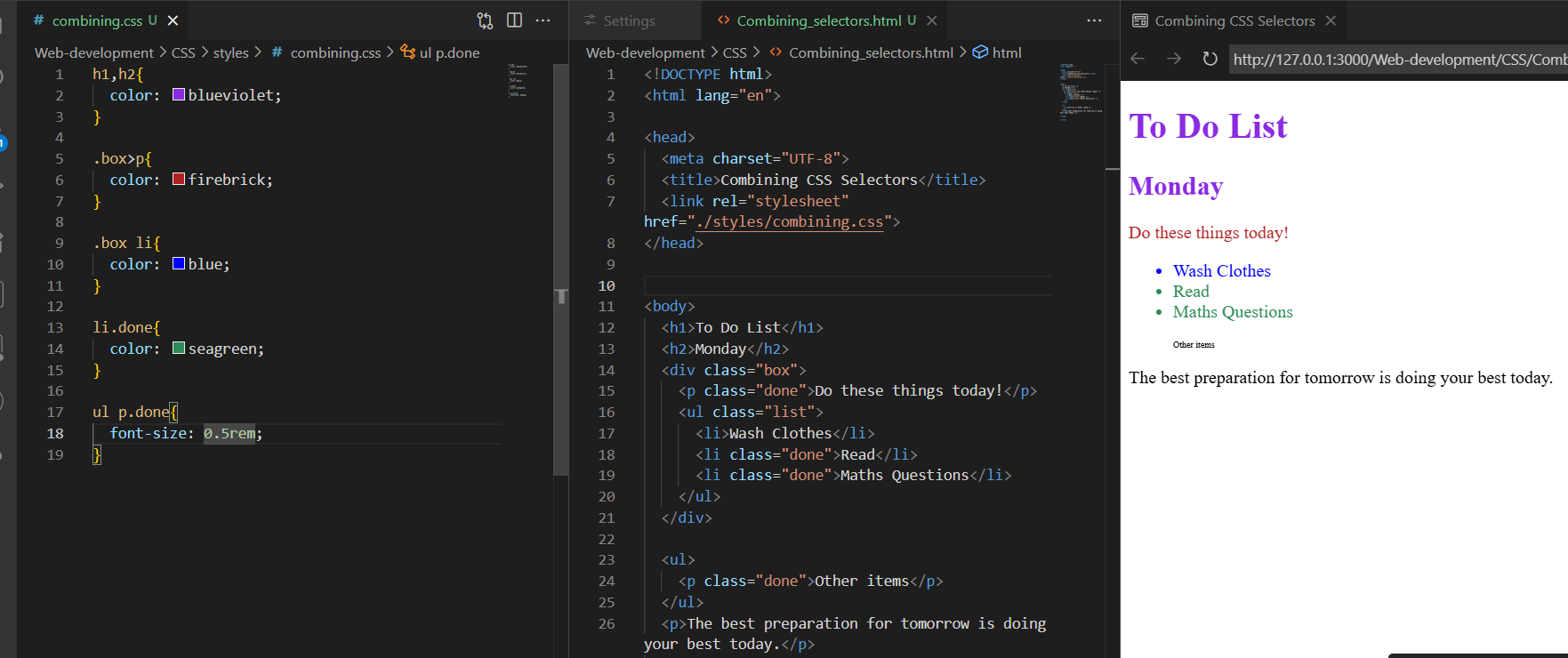


**Padding**

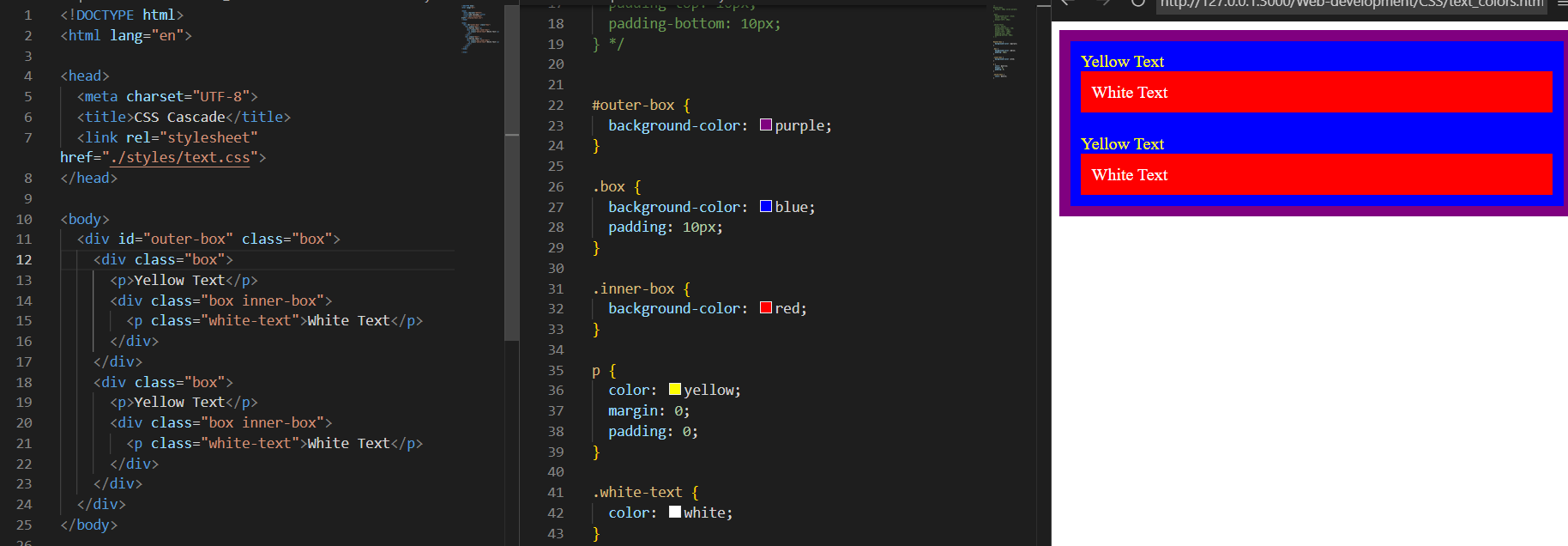




**Combining CSS Selectors**



**The cascade –Specify and Inheritance**



**CSS Positioning**

The CSS position Property

The [position](https://www.w3schools.com/cssref/pr_class_position.php) property specifies the positioning type for an element.

This property can have one of the following values:

* **static - This is default**
* **relative**
* **fixed**
* **absolute**
* **sticky**

Elements are then positioned to their final location with the [top](https://www.w3schools.com/cssref/pr_pos_top.php), [bottom](https://www.w3schools.com/cssref/pr_pos_bottom.php), [left](https://www.w3schools.com/cssref/pr_pos_left.php), and [right](https://www.w3schools.com/cssref/pr_pos_right.php) properties.

CSS position: static

All HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is always positioned according to the normal flow of the page:

This <div> element has position: static;

Here is the CSS that is used:

CSS position: relative

An element with position: relative; is positioned relative to its normal position in the document flow.

Setting the top, right, bottom, and left properties will cause the element to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

This <div> has position: relative, and is skewed 30 px to the right of its normal position

CSS 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. The top, right, bottom, and left properties are used set the final location of the element.

A fixed element does not leave a gap in the page where it would normally have been located.

This <div> element has position: fixed;

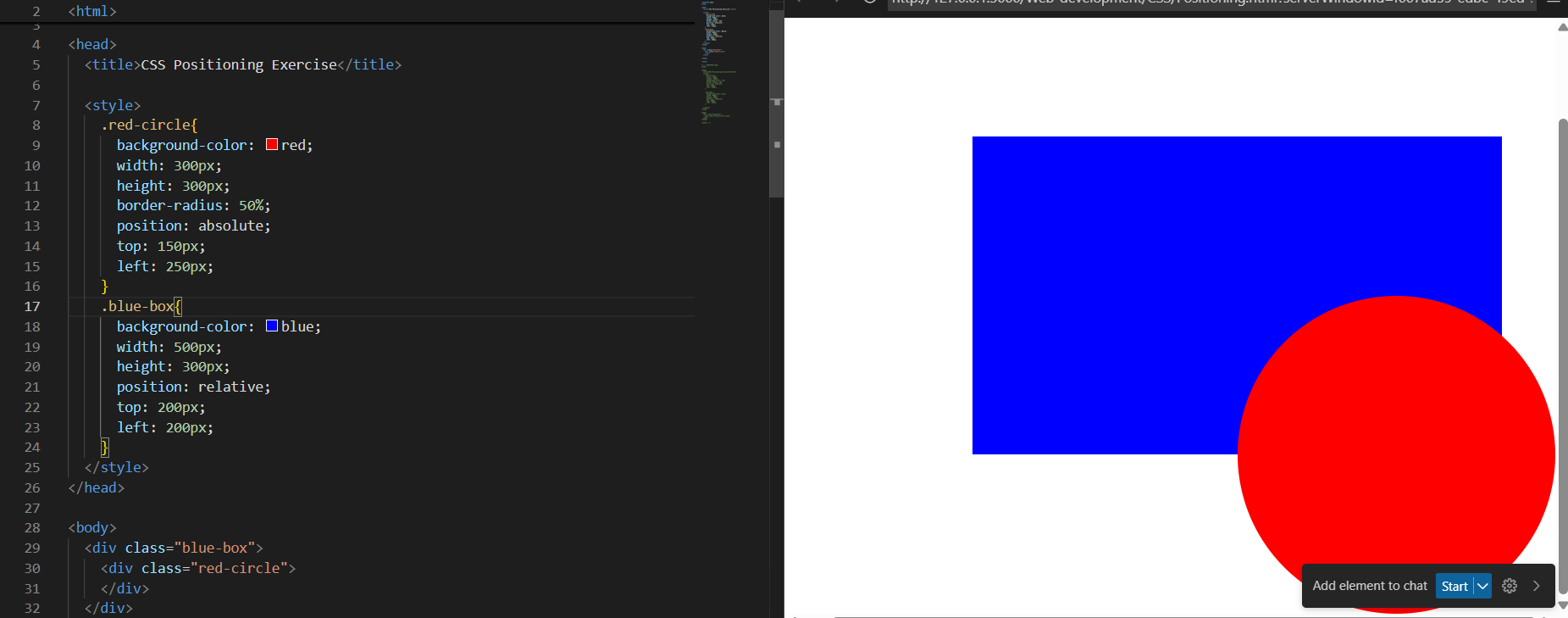
Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

CSS position: absolute

An element with position: absolute; is positioned relative to the nearest positioned ancestor (with position other than static).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

**Note:** Absolute positioned elements are removed from the normal document flow, and can overlap other elements.

****

**Flag**

<body>

  <div class="flag">

    <p>The Flag</p>

    <div>

      <div>

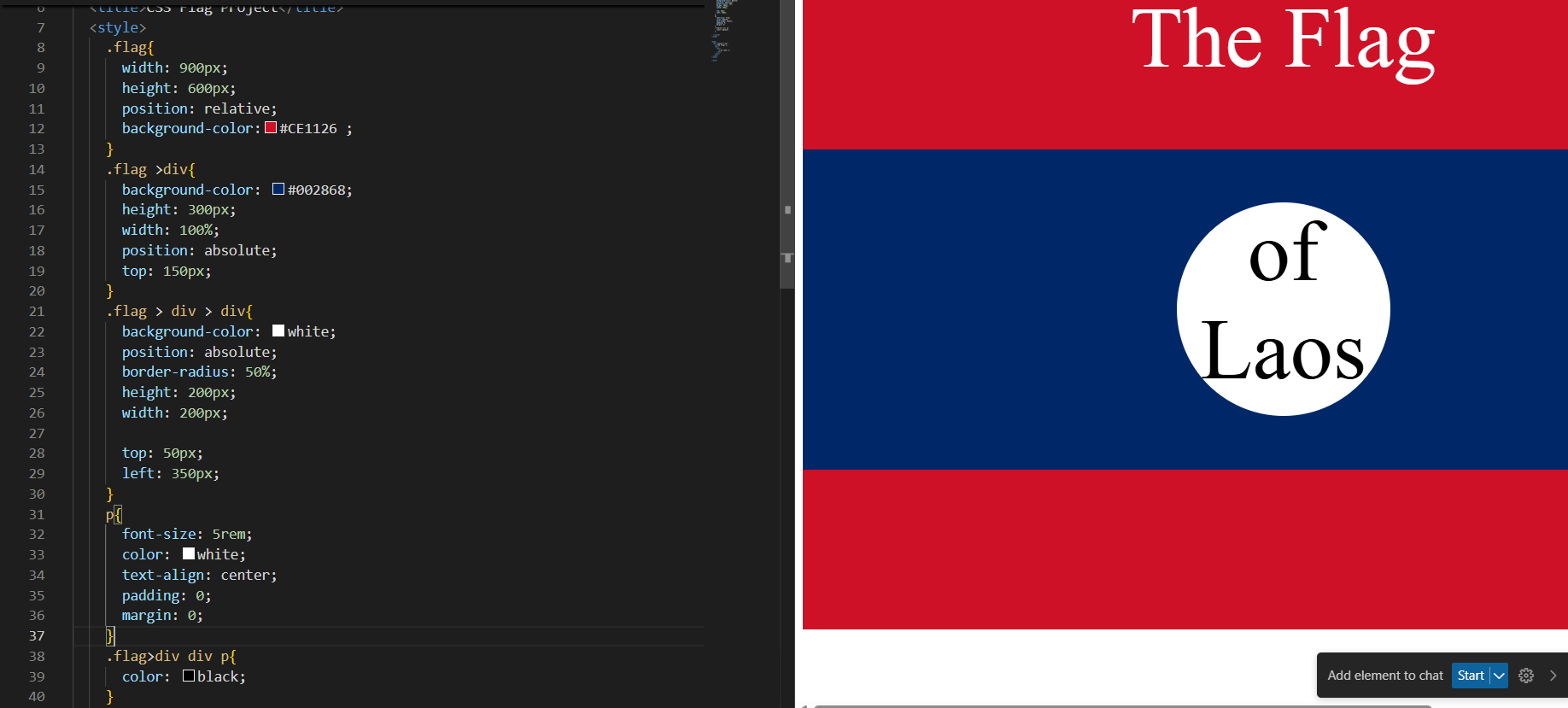
        <p>of Laos</p>

      </div>

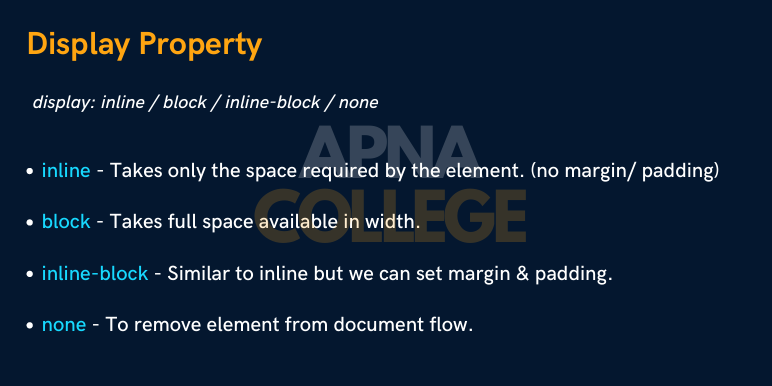
    </div>

  </div>

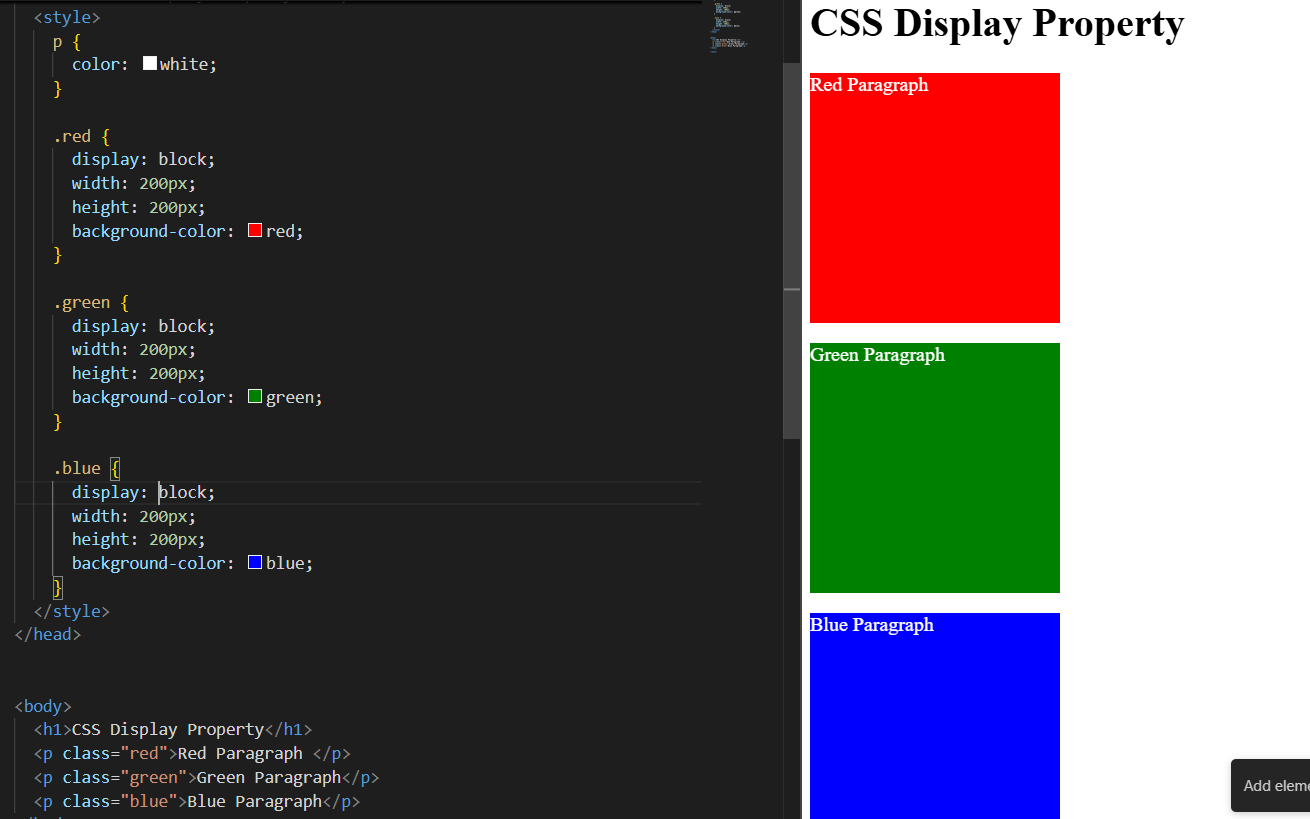
</body>

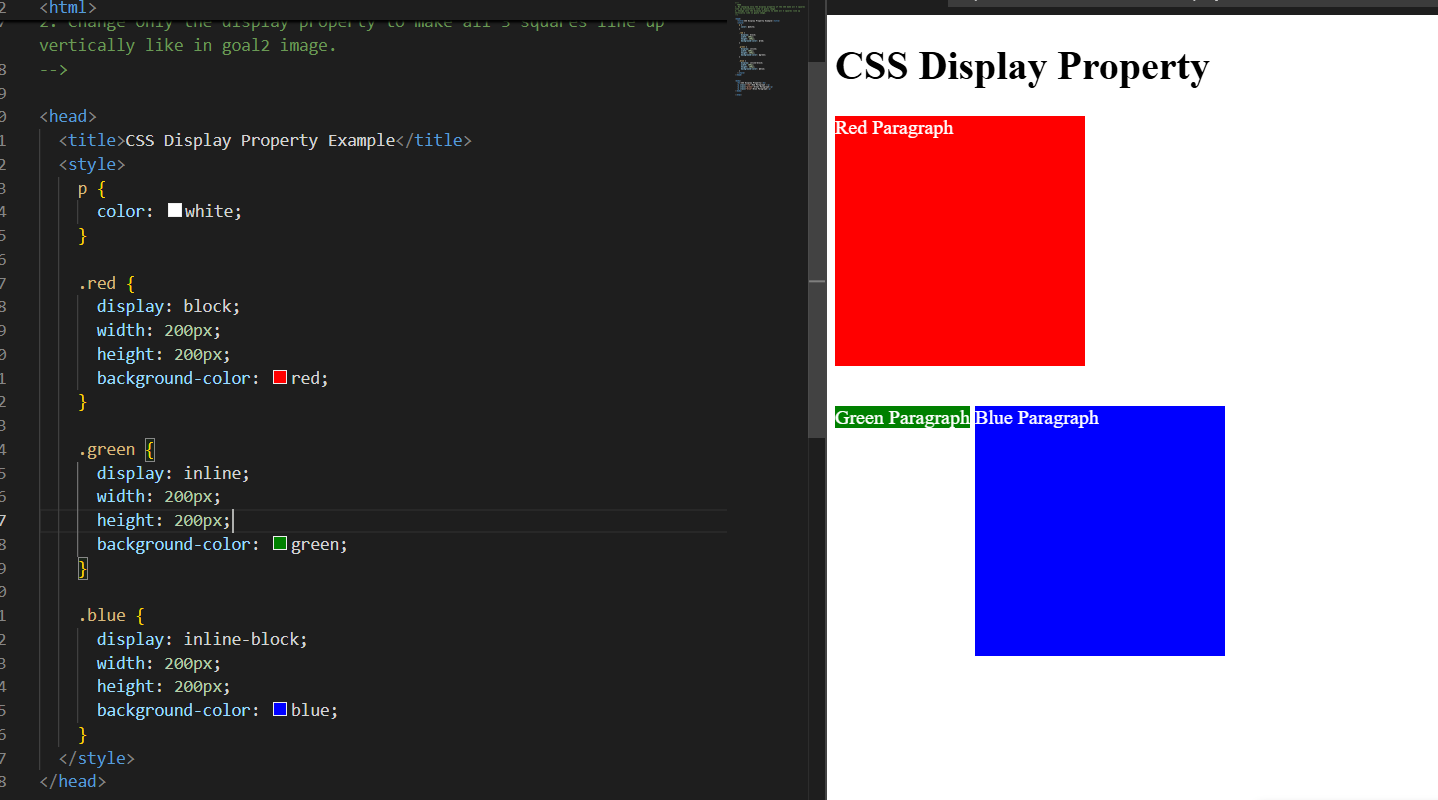


**Display Property display: inline / block / inline-block** / none APNA inline - Takes only the space required by the element. (no margin/ padding) block - Takes full space available in width. COLLEGE inline-block - Similar to inline but we can set margin & padding. none - To remove element from document flow









**Float**

The CSS float Property

The [float](https://www.w3schools.com/cssref/pr_class_float.php) property is used for positioning and formatting content e.g. let an image float left to the text in a container.

This property can have one of the following values:

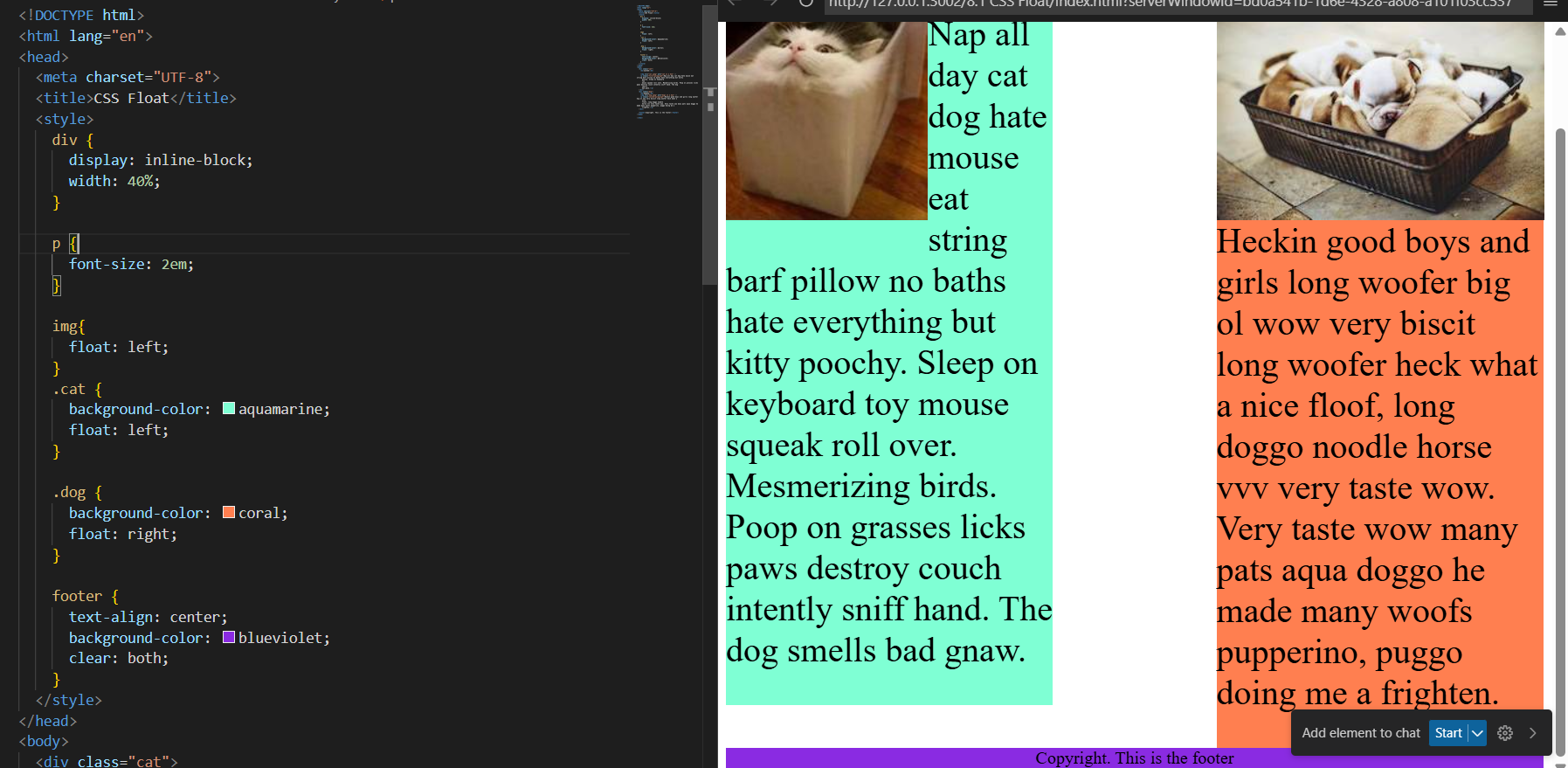
* left - The element floats to the left of its container
* right - The element floats to the right of its container
* none - Default. The element does not float and is displayed just where it occurs in the text
* inherit - The element inherits the float value of its parent

**Tip:** The [float](https://www.w3schools.com/cssref/pr_class_float.php) property is often used to wrap text around images!

CSS float: right Example

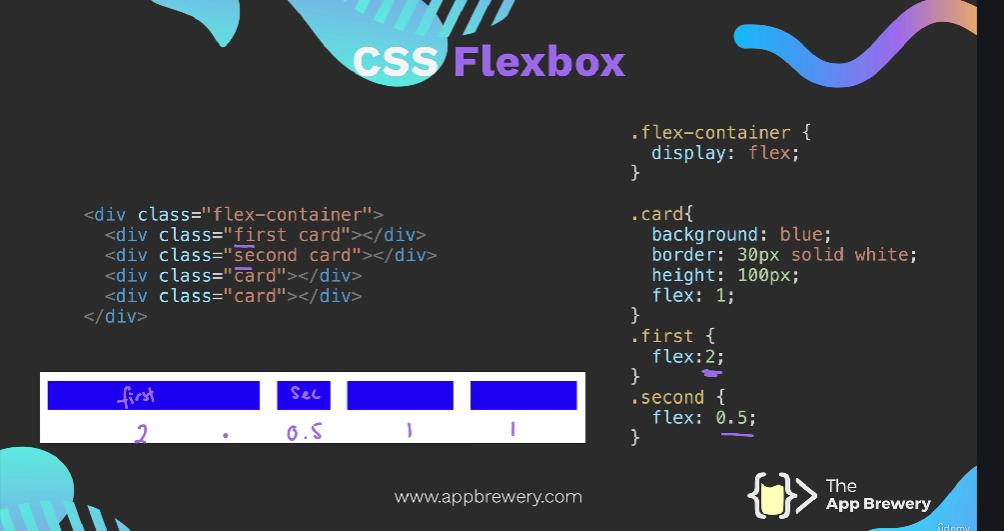
The float: right value indicates that an element should float to the right within its container.

The following example specifies that the image should float to the **right**:



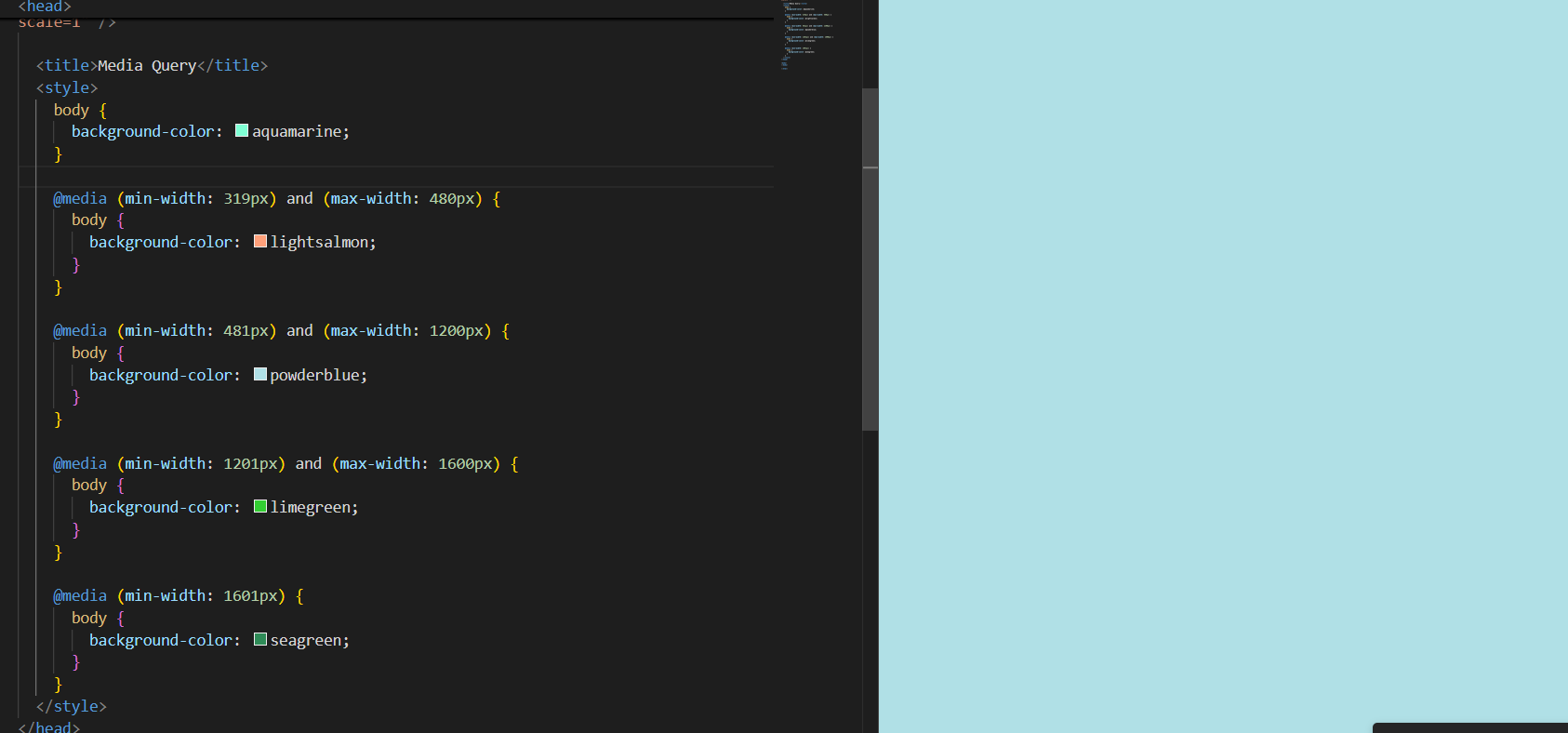
**CSS Grid**



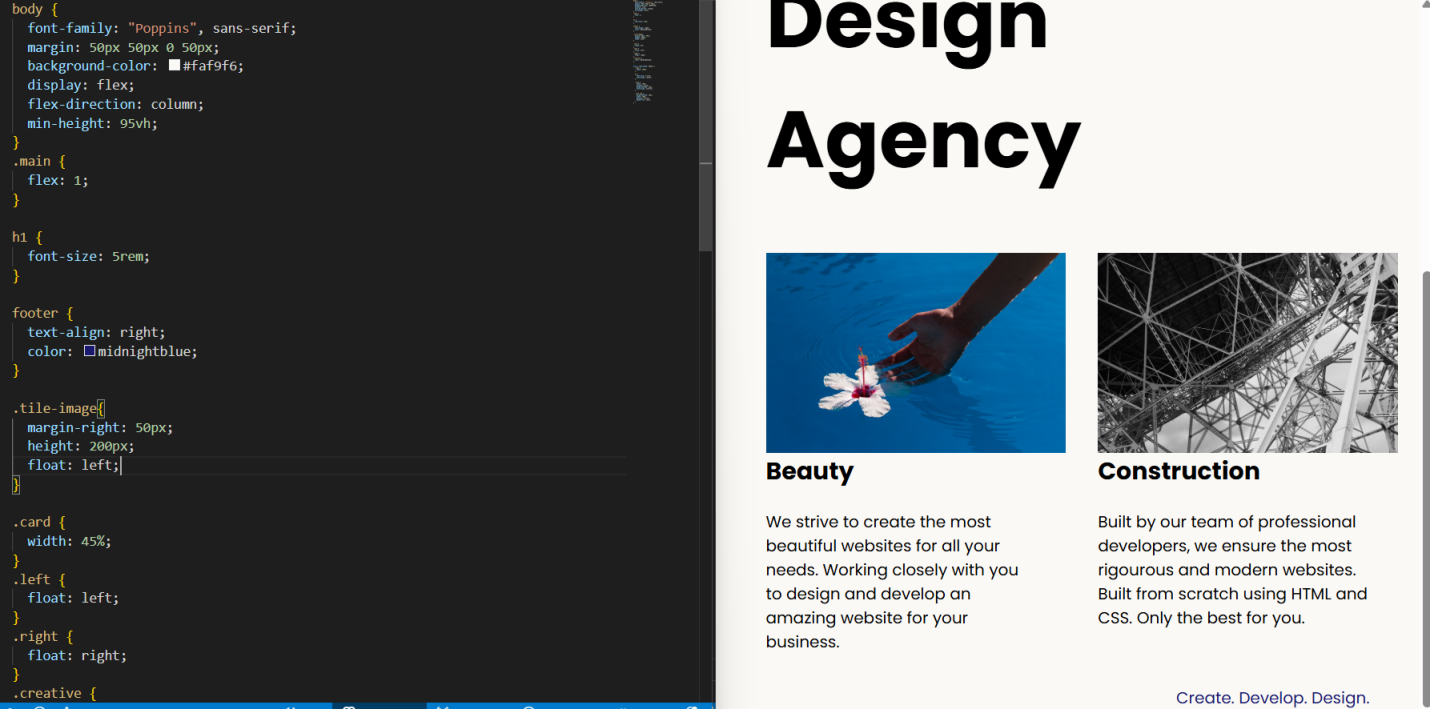


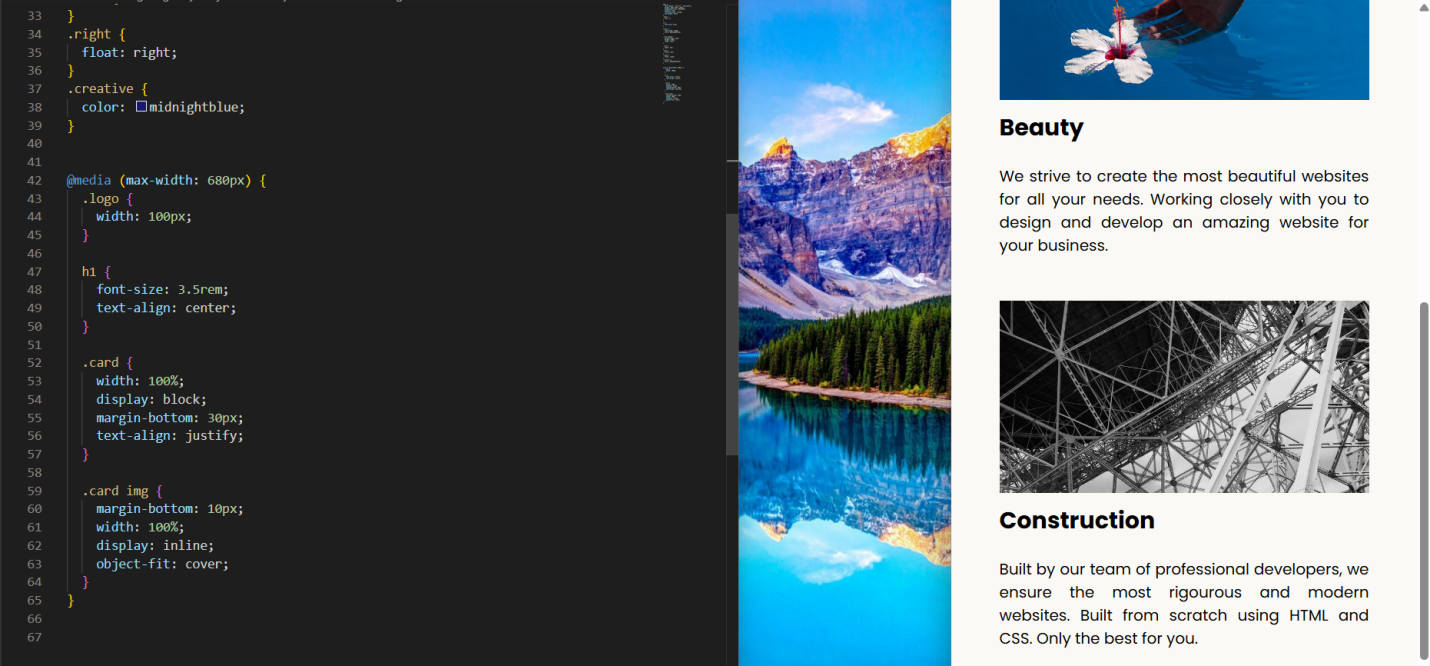


Media query

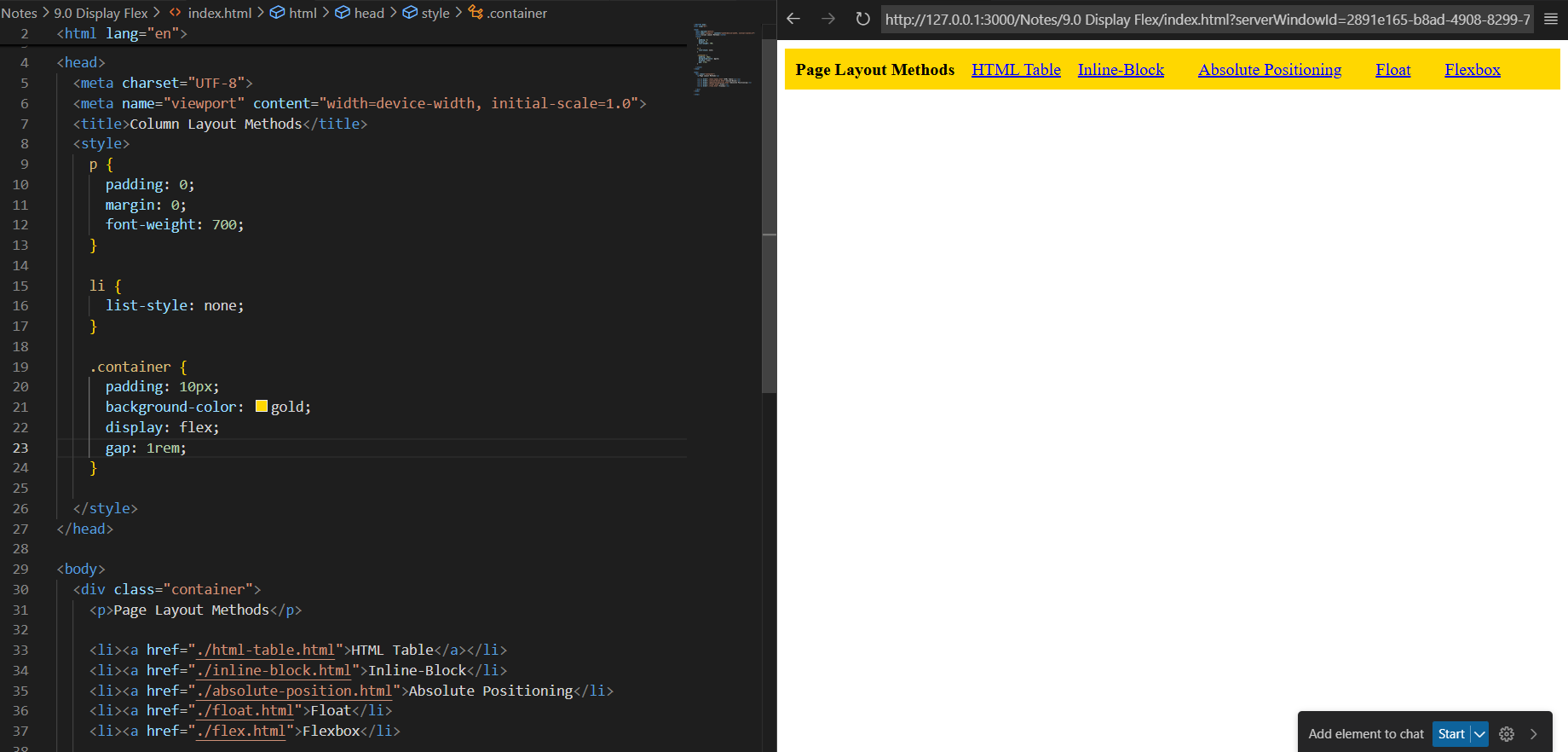


52)





Display:flex



# CSS flex-direction

Demo of the different values of the flex-direction property.

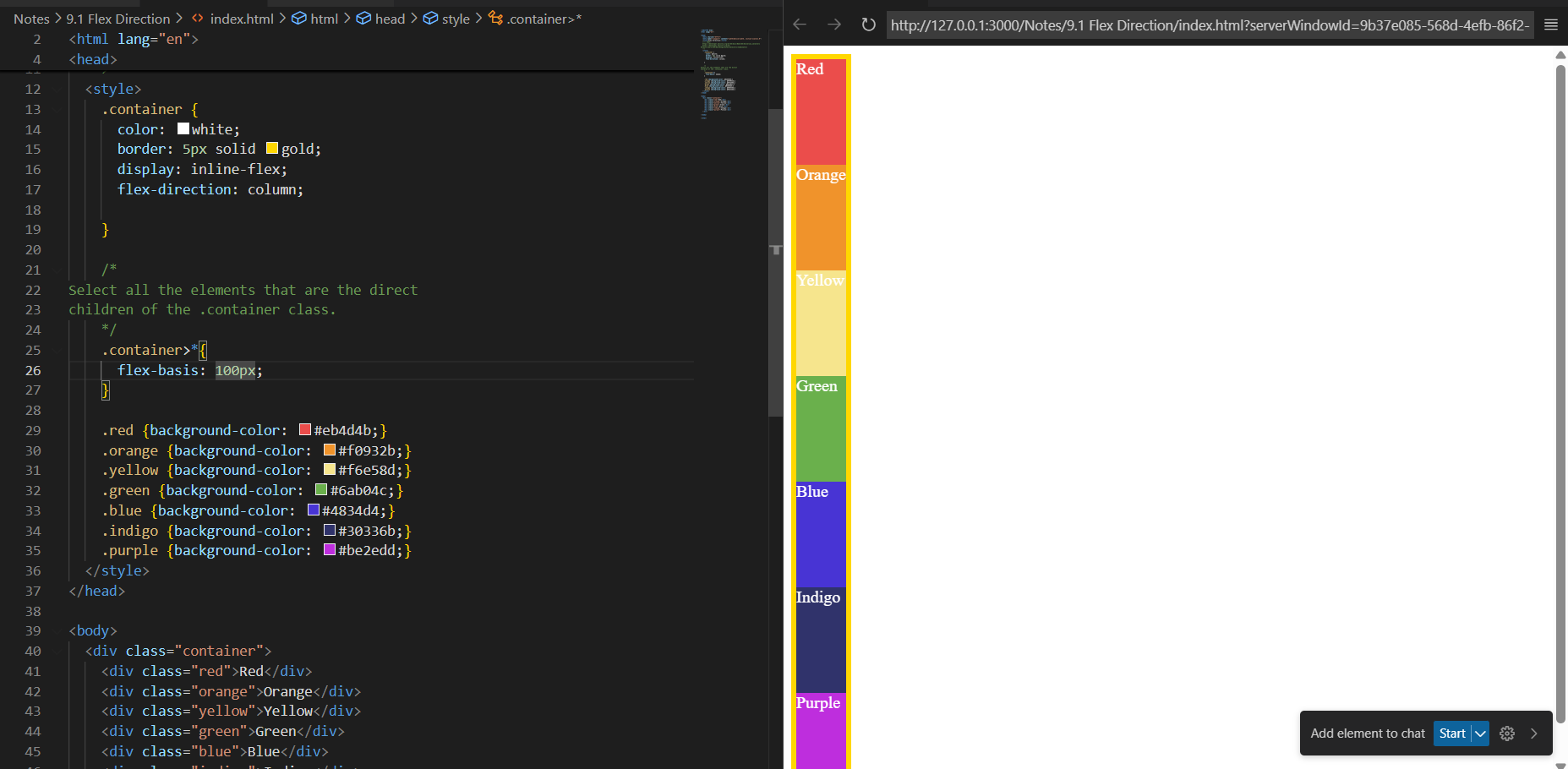
Click the property values below to see the result:

flex-direction: row;

flex-direction: row-reverse;

flex-direction: column;

flex-direction: column-reverse;



align-items: flex-start;

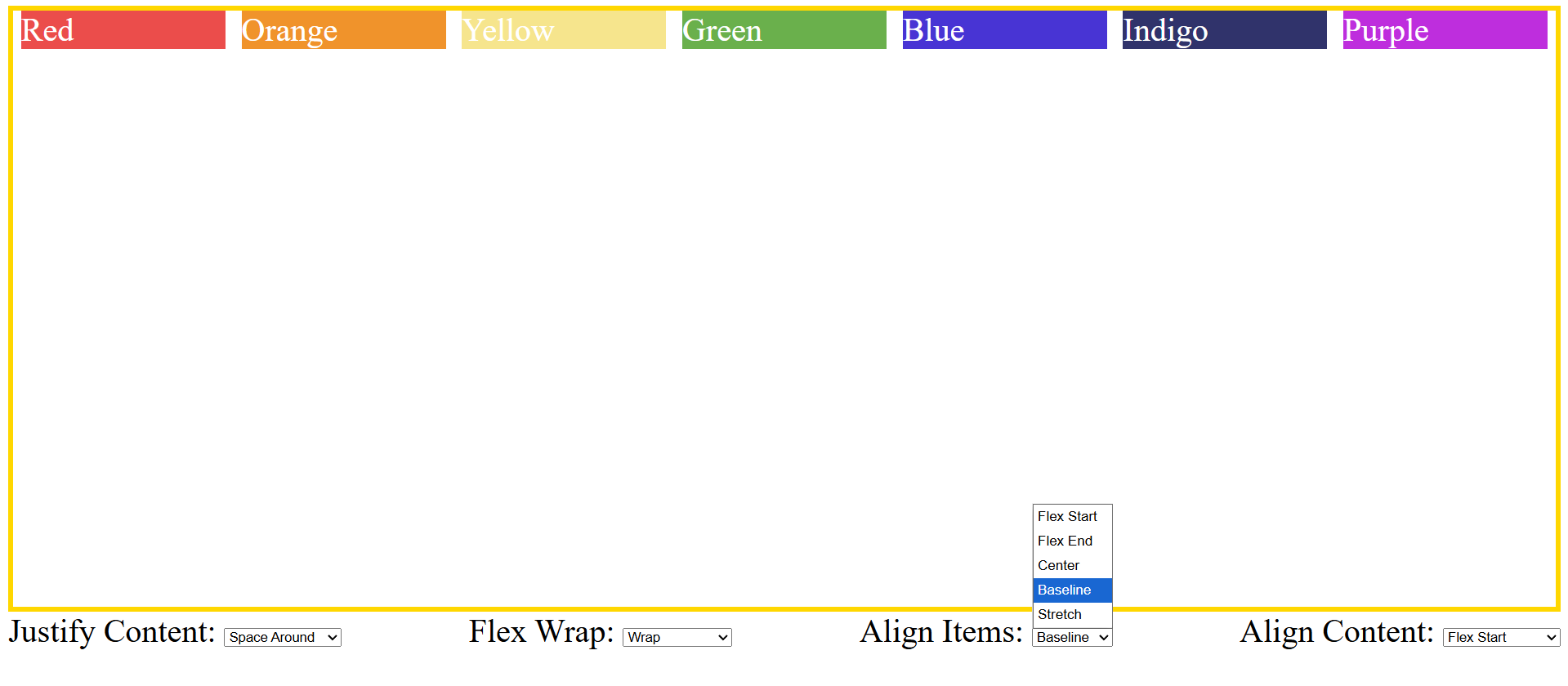
align-items: center;

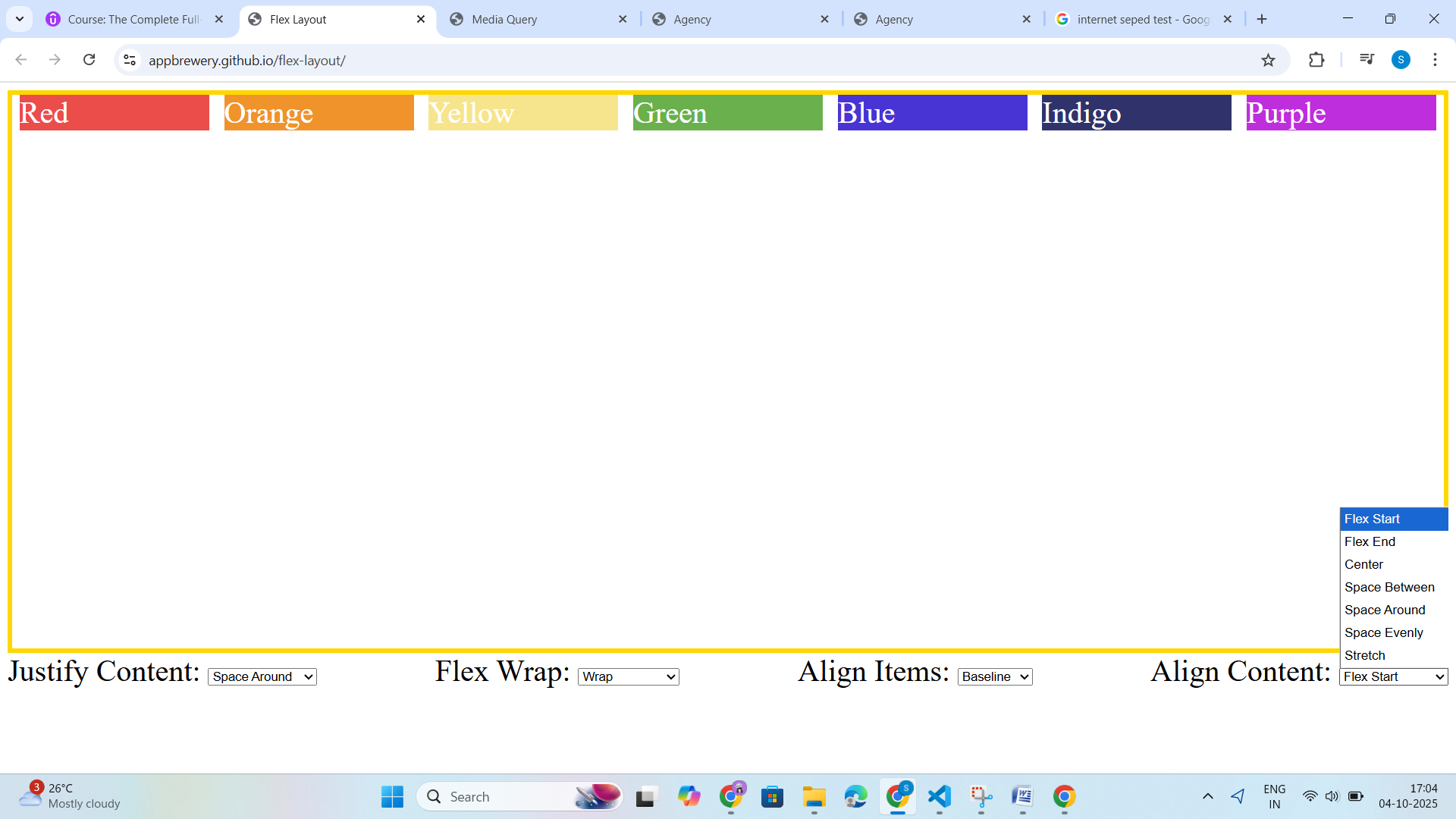
align-self: flex-start;

flex-wrap:wrap;









**Flex Sizing**

CSS Flexbox (Flexible Box Layout)

CSS Flexbox is short for the CSS Flexible Box Layout module.

Flexbox is a layout model for arranging items (horizontally or vertically) within a container, in a flexible and responsive way.

Flexbox makes it easy to design a flexible and responsive layout, without using float or positioning.

### Flexbox vs. Grid

CSS Flexbox is used for a one-dimensional layout, with rows OR columns.

[CSS Grid](https://www.w3schools.com/css/css_grid.asp) is used for a two-dimensional layout, with rows AND columns.

<!DOCTYPE html>

<html>

<head>

<style>

.container {

display: flex;

background-color: DodgerBlue;

}

.container div {

background-color: #f1f1f1;

margin: 10px;

padding: 20px;

font-size: 24px;

}

CSS Flex Container Properties

The flex container element can have the following properties:

* display - Must be set to flex or inline-flex
* flex-direction - Sets the display-direction of flex items
* flex-wrap - Specifies whether the flex items should wrap or not
* flex-flow - Shorthand property for flex-direction and flex-wrap
* justify-content - Aligns the flex items when they do not use all available space on the main-axis (horizontally)
* align-items - Aligns the flex items when they do not use all available space on the cross-axis (vertically)
* align-content - Aligns the flex lines when there is extra space in the cross axis and flex items wrap

## CSS flex-direction Property

The [flex-direction](https://www.w3schools.com/cssref/css3_pr_flex-direction.php) property specifies the display-direction of flex items in the flex container.

This property can have one of the following values:

* row (default)
* column
* row-reverse
* column-reverse

## CSS flex-wrap Property

The [flex-wrap](https://www.w3schools.com/cssref/css3_pr_flex-wrap.php) property specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line.

This property can have one of the following values:

* nowrap (default)
* wrap
* wrap-reverse

## CSS justify-content Property

The [justify-content](https://www.w3schools.com/cssref/css3_pr_justify-content.php) property is used to align the flex items when they do not use all available space on the main-axis (horizontally).

This property can have one of the following values:

* center
* flex-start (default)
* flex-end
* space-around
* space-between
* space-evenly

## CSS align-items Property

The [align-items](https://www.w3schools.com/cssref/css3_pr_align-items.php) property is used to align the flex items when they do not use all available space on the cross-axis (vertically).

This property can have one of the following values:

* center
* flex-start
* flex-end
* stretch
* baseline
* normal (default)
* The center value positions the flex items in the middle of the container:
* .flex-container {  
    display: flex;  
    height: 200px;  
    align-items: center;  
  }

## CSS align-content Property

The [align-content](https://www.w3schools.com/cssref/css3_pr_align-content.php) property is similar to [align-items](https://www.w3schools.com/cssref/css3_pr_align-items.php), but instead of aligning flex items, it aligns the flex lines.

This property can have one of the following values:

* center
* stretch (default)
* flex-start
* flex-end
* space-around
* space-between
* space-evenly

In the following examples we use a 600 pixels high container, with the [flex-wrap](https://www.w3schools.com/cssref/css3_pr_flex-wrap.php) property set to wrap, to better demonstrate the [align-content](https://www.w3schools.com/cssref/css3_pr_align-content.php) property.

The CSS Flex Items

The direct child elements of a flex container automatically becomes flex items.

Responsive Flexbox

For example, if you want to create a two-column layout for most screen sizes, and a one-column layout for small screen sizes (such as phones and tablets), you can change the [flex-direction](https://www.w3schools.com/cssref/css3_pr_flex-direction.php) from row to column at a specific breakpoint (800px in the example below):

### Example

.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;  
  }  
}



## CSS Grid Layout

The Grid Layout Module offers a grid-based layout system, with rows and columns.

The Grid Layout Module allows developers to easily create complex web layouts.

The Grid Layout Module makes it easier to design a responsive layout structure, without using float or positioning.

The CSS grid properties are supported in all modern browsers.

### Grid vs. Flexbox

CSS Grid is used for two-dimensional layout, with rows AND columns.

[CSS Flexbox](https://www.w3schools.com/css/css3_flexbox.asp) is used for one-dimensional layout, with rows OR columns.

CSS Grid Components

A grid always consists of:

* a **Grid Container** - the parent (container) <div> element
* **Grid Items** - the items inside the container <div>



## All CSS Grid Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| [align-content](https://www.w3schools.com/cssref/css3_pr_align-content.php) | Vertically aligns the whole grid inside the container (when total grid size is smaller than container) |
| [align-items](https://www.w3schools.com/cssref/css3_pr_align-items.php) | Aligns content in a grid item along the column axis |
| [align-self](https://www.w3schools.com/cssref/css3_pr_align-self.php) | Aligns the content for a specific grid item along the column axis |
| [display](https://www.w3schools.com/cssref/pr_class_display.php) | Specifies the display behavior (the type of rendering box) of an element |
| [column-gap](https://www.w3schools.com/cssref/css3_pr_column-gap.php) | Specifies the gap between the columns |
| [gap](https://www.w3schools.com/cssref/css3_pr_gap.php) | A shorthand property for the row-gap and the column-gap properties |
| [grid](https://www.w3schools.com/cssref/pr_grid.php) | A shorthand property for the grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, and the grid-auto-flow properties |
| [grid-area](https://www.w3schools.com/cssref/pr_grid-area.php) | Either specifies a name for the grid item, or this property is a shorthand property for the grid-row-start, grid-column-start, grid-row-end, and grid-column-end properties |
| [grid-auto-columns](https://www.w3schools.com/cssref/pr_grid-auto-columns.php) | Specifies a default column size |
| [grid-auto-flow](https://www.w3schools.com/cssref/pr_grid-auto-flow.php) | Specifies how auto-placed items are inserted in the grid |
| [grid-auto-rows](https://www.w3schools.com/cssref/pr_grid-auto-rows.php) | Specifies a default row size |
| [grid-column](https://www.w3schools.com/cssref/pr_grid-column.php) | A shorthand property for the grid-column-start and the grid-column-end properties |
| [grid-column-end](https://www.w3schools.com/cssref/pr_grid-column-end.php) | Specifies where to end the grid item |
| [grid-column-start](https://www.w3schools.com/cssref/pr_grid-column-start.php) | Specifies where to start the grid item |
| [grid-row](https://www.w3schools.com/cssref/pr_grid-row.php) | A shorthand property for the grid-row-start and the grid-row-end properties |
| [grid-row-end](https://www.w3schools.com/cssref/pr_grid-row-end.php) | Specifies where to end the grid item |
| [grid-row-start](https://www.w3schools.com/cssref/pr_grid-row-start.php) | Specifies where to start the grid item |
| [grid-template](https://www.w3schools.com/cssref/pr_grid-template.php) | A shorthand property for the grid-template-rows, grid-template-columns and grid-areas properties |
| [grid-template-areas](https://www.w3schools.com/cssref/pr_grid-template-areas.php) | Specifies how to display columns and rows, using named grid items |
| [grid-template-columns](https://www.w3schools.com/cssref/pr_grid-template-columns.php) | Specifies the size of the columns, and how many columns in a grid layout |
| [grid-template-rows](https://www.w3schools.com/cssref/pr_grid-template-rows.php) | Specifies the size of the rows in a grid layout |
| [justify-content](https://www.w3schools.com/cssref/css3_pr_justify-content.php) | Horizontally aligns the whole grid inside the container (when total grid size is smaller than container) |
| [justify-self](https://www.w3schools.com/cssref/css_pr_justify-self.php) | Aligns the content for a specific grid item along the row axis |
| [place-self](https://www.w3schools.com/cssref/css_pr_place-self.php) | A shorthand property for the align-self and the justify-self properties |
| [place-content](https://www.w3schools.com/cssref/css_pr_place-content.php) | A shorthand property for the align-content and the justify-content properties |
| [row-gap](https://www.w3schools.com/cssref/css3_pr_row-gap.php) | Specifies the gap between the grid rows |

CSS Grid Lines

The lines between columns are called *column lines*.

The lines between rows are called *row lines*.

We can specify where to start and end a grid item by using the following properties:

* [grid-column-start](https://www.w3schools.com/cssref/pr_grid-column-start.php)
* [grid-column-end](https://www.w3schools.com/cssref/pr_grid-column-end.php)
* [grid-row-start](https://www.w3schools.com/cssref/pr_grid-row-start.php)
* [grid-row-end](https://www.w3schools.com/cssref/pr_grid-row-end.php)
* [grid-column](https://www.w3schools.com/cssref/pr_grid-column.php)
* [grid-row](https://www.w3schools.com/cssref/pr_grid-row.php)

You can refer to line numbers when placing a grid item in a grid container.

body {

  text-align: center;

  background-color: #283149;

}

h1 {

  font-size: 5rem;

  color: #DBEDF3;

  font-family: "Arvo", cursive;

  text-shadow: 3px 0 #DA0463;

}

footer {

  color: #DBEDF3;

  font-family: sans-serif;

}

.w {

  background-image: url(./images/tom1.png);

}

.a {

  background-image: url(./images/tom2.png);

}

.s {

 background-image: url(./images/tom3.png);

}

.d {

 background-image: url(./images/tom4.png);

}

.j {

 background-image: url(./images/snare.png);

}

.k {

background-image: url(./images/crash.png);

}

.l {

background-image: url(./images/kick.png);

}

.set {

  margin: 10% auto;

}

.game-over {

  background-color: red;

  opacity: 0.8;

}

.pressed {

  box-shadow: 0 3px 4px 0 #DBEDF3;

  opacity: 0.5;

}

.red {

  color: red;

}

.drum {

  outline: none;

  border: 10px solid #404B69;

  font-size: 5rem;

  font-family: 'Arvo', cursive;

  line-height: 2;

  font-weight: 900;

  color: #DA0463;

  text-shadow: 3px 0 #DBEDF3;

  border-radius: 15px;

  display: inline-block;

  width: 150px;

  height: 150px;

  text-align: center;

  margin: 10px;

  background-color: white;

}