

# CSS

```
</header>  
< background-color: #3498db,  
color: #fff;  
padding: 20px; text-align: center; text-align: center; text-align: center;  
>/>  
<< /header>
```

CSS



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

```
><div class="container">  
</div>
```

Sample Text

```
.container {  
< padding: 20px;  
< margin: 0 auto;  
< max-width: 1200px;  
< text-align: center;  
>}
```



# Table of Contents

<b>Introduction</b> .....	<b>1</b>
Purpose .....	1
Scope .....	1
Procedure .....	1
Brief Problem .....	1
<b>CSS Intro &amp; Syntax</b> .....	<b>2</b>
CSS Syntax .....	2
Types of CSS .....	4
1. Inline CSS .....	4
2. Internal CSS .....	5
3. External CSS .....	6
CSS Priority (Cascade Order) .....	7
<b>CSS Selectors</b> .....	<b>8</b>
1. Element (Type) Selector .....	8
2. Class Selector .....	8
3. ID Selector .....	8
4. Universal Selector .....	9
5. Grouping Selector .....	9
6. Attribute Selectors .....	9
6.1 Attribute Exists Selector .....	9
6.2 Exact Attribute Value Selector .....	9
6.3 Attribute Contains Selector .....	9
6.4 Attribute Starts With .....	10
6.5 Attribute Ends With .....	10
7. Combinator Selectors .....	10
7.1 Descendant Selector (space) .....	10
7.2 Child Selector (>) .....	10
7.3 Adjacent Sibling Selector (+) .....	10
7.4 General Sibling Selector (~) .....	11
8. Pseudo-Classes .....	11
8. Pseudo-Elements .....	13
8.1 First Letter .....	13
8.2 First Line .....	13
8.3 Before & After .....	13

10. Selector Priority (Specificity).....	13
<b>Fonts &amp; Text.....</b>	<b>14</b>
FONT-FAMILY .....	14
FONT-SIZE .....	15
FONT-WEIGHT .....	16
FONT-STYLE .....	17
FONT-VARIANT .....	18
LINE-HEIGHT.....	19
COLOR .....	20
TEXT-ALIGN.....	21
TEXT-DECORATION .....	22
TEXT-TRANSFORM.....	23
LETTER-SPACING & WORD-SPACING.....	23
WHITE-SPACE .....	24
TEXT-OVERFLOW .....	25
TEXT-INDENT .....	26
TEXT-SHADOW .....	27
WRITING-MODE.....	28
TEXT-ORIENTATION .....	29
Web Fonts .....	30
@FONT-FACE .....	30
BASIC @FONT-FACE SYNTAX .....	30
FONT FORMATS (IMPORTANT).....	31
MULTIPLE FONT WEIGHTS .....	31
MULTIPLE STYLES (ITALIC) .....	31
FONT-DISPLAY (VERY IMPORTANT) .....	32
USING GOOGLE FONTS .....	32
Option 1 — <link> (most common) .....	32
Option 2 — @import (not recommended -> slower than link).....	32
<b>CSS Box Model.....</b>	<b>33</b>
Content.....	33
Padding.....	34
Padding properties:.....	34
Padding values: .....	34
Border .....	35

Border properties:.....	35
Common border styles:.....	35
Border Radius.....	36
Properties & syntax:.....	36
Possible values: .....	36
Border Image.....	37
Required setup:.....	37
Main Properties .....	37
Common values: .....	37
Important notes: .....	37
Margin .....	39
Margin properties: .....	39
Margin values:.....	39
Box Sizing .....	40
Box Shadow.....	41
Syntax:.....	41
Properties & values:.....	41
<b>CSS Layout &amp; Flow .....</b>	<b>42</b>
DISPLAY .....	42
What display controls .....	42
Common values.....	42
VISIBILITY.....	44
POSITIONING.....	45
Position types.....	45
Static vs Relative .....	45
Absolute with Positioned Parent .....	46
Fixed Position .....	47
Z-INDEX & STACKING CONTEXT.....	48
Rules.....	48
FLOAT & CLEAR.....	49
Float behavior .....	49
Clear behavior .....	49
<b>CSS Flexbox.....</b>	<b>50</b>
Base Setup (Shared by All Examples) .....	50
flex-direction .....	51

flex-wrap .....	52
justify-content.....	53
align-content (requires wrap) .....	54
align-items.....	56
align-self & justify-self.....	57
Flex Item Properties (grow, shrink, basis, order) .....	57
flex-grow .....	57
flex-shrink.....	58
flex-basis.....	59
order.....	60
<b>CSS Grid Layout .....</b>	<b>61</b>
Grid Basics (must exist first).....	61
grid-template-columns .....	61
grid-template-rows .....	62
gap (row-gap / column-gap).....	63
Grid Alignment (Container).....	64
Grid Content Alignment (whole grid).....	64
Grid Item Alignment (per item) .....	64
Grid Lines (Positioning by lines) .....	65
Grid Areas (named layouts) .....	66
<b>CSS Backgrounds .....</b>	<b>67</b>
background-color .....	67
background-image .....	68
background-repeat .....	69
background-position .....	70
background-size .....	71
background-clip & background-origin .....	72
background-attachment .....	73
Overflow.....	74
<b>Filters .....</b>	<b>75</b>
<b>CSS Transforms, Transitions &amp; Animations .....</b>	<b>77</b>
Transforms.....	77
transform .....	77
Possible functions .....	77
transform-origin.....	77

3D Transforms .....	79
perspective.....	79
perspective-origin .....	79
transform-style.....	80
backface-visibility.....	80
Transitions .....	82
transition .....	82
transition-property .....	82
transition-duration.....	82
transition-timing-function.....	83
transition-delay .....	83
Animations .....	84
@keyframes .....	84
Animation Properties.....	84
<b>Responsive Design .....</b>	<b>85</b>
Core Principles of Responsive Design .....	85
Viewport.....	85
Responsive Units.....	86
Responsive Images.....	86
Media Queries.....	86
Breakpoints (Common Practice) .....	87
Responsive Typography.....	87
Orientation.....	87
Hiding & Showing Content .....	87
<b>Conclusion.....</b>	<b>88</b>
<b>Feedback &amp; Contribution .....</b>	<b>88</b>
<b>Copyright &amp; Usage.....</b>	<b>88</b>
<b>Acknowledgments .....</b>	<b>89</b>

# Introduction

## Purpose

The purpose of this document is to provide a **comprehensive guide to CSS (Cascading Style Sheets)**, covering all essential concepts, properties, and techniques required to design and style modern web pages. This document aims to serve both as a **learning resource for beginners** and a **reference manual for intermediate developers**, detailing the usage, values, and practical implementation of CSS properties.

## Scope

This document covers a wide range of CSS topics, including **selectors, pseudo-classes, box model, positioning, display types, flexbox, grid, backgrounds, typography, transitions, animations, filters, and responsive design**. Each section is accompanied by **examples, visual demonstrations, and explanations**, allowing the reader to understand the effect of each property in real-world layouts. The guide is structured to progress from **basic concepts to advanced techniques**, ensuring a clear learning path.

## Procedure

The procedure followed in this document involves:

1. **Introduction of concepts** – explaining each property and its purpose.
2. **Listing possible values** – providing all common and advanced options.
3. **Practical examples** – using colored, numbered boxes, or live code snippets to visually demonstrate each property's effect.
4. **Comparison and clarification** – highlighting differences between similar properties or values, such as align-items vs align-self.

## Brief Problem

Many developers and students face **difficulty visualizing the effects of CSS properties**, especially when properties interact or override each other. The cascading nature, specificity, and layout differences between Flexbox and Grid often lead to confusion. This document addresses these challenges by providing **clear, complete, and visually demonstrative examples** that allow learners to see the impact of each CSS property in isolation and in combination.

# CSS Intro & Syntax

**CSS (Cascading Style Sheets)** is the language used to style and visually format HTML documents. While **HTML** defines the structure and content of a webpage (headings, paragraphs, images, buttons), **CSS** controls how that content looks — colors, fonts, spacing, layout, responsiveness, and animations.

---

In simple terms:

- **HTML = What is on the page**
  - **CSS = How it looks**
- 

CSS allows developers to:

- Separate content from design
  - Reuse styles across multiple pages
  - Make websites responsive on different screen sizes
  - Improve user experience and accessibility
- 

## CSS Syntax

A CSS rule is made of **three main parts**:

```
selector {  
  property: value;  
}
```

- **Selector** → selects the HTML element to style
- **Property** → what you want to change
- **Value** → how you want to change it

### Example:

```
p {  
color: blue;  
font-size: 16px;  
}
```

### This rule:

- Targets all <p> (paragraph) elements
- Changes the text color to blue
- Sets the font size to 16 pixels

Every property-value pair must end with a **semicolon (;)**.

# Types of CSS

## 1. Inline CSS

Inline CSS is written **directly inside an HTML element** using the style attribute.

### Example:

```
<p style="color: red; font-size: 18px;">This is inline CSS</p>
```

### Characteristics:

- Applied to **one element only**
- Has the **highest priority**

## 2. Internal CSS

Internal CSS is written inside a <style> tag placed in the <head> section of an HTML document.

### Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
color: green;
font-size: 16px;
}
</style>
</head>
<body>
<p>This is internal CSS</p>
</body>
</html>
```

### Characteristics:

- Affects **only one HTML page**

### 3. External CSS

External CSS is written in a **separate .css file** and linked to HTML using the <link> tag.

#### CSS File (styles.css):

```
p {  
color: blue;  
font-size: 16px;  
}
```

#### HTML File:

```
<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" href="styles.css">  
</head>  
<body>  
<p>This is external CSS</p>  
</body>  
</html>
```

#### Characteristics:

- Best practice
- Styles multiple pages at once

## *CSS Priority (Cascade Order)*

When **multiple CSS rules** target the same element, CSS follows a priority order:

- **Inline CSS** (highest priority, performed last)
- **Internal CSS**
- **External CSS** (lowest priority, performed first)
  
- If two rules have the **same type**, the **last one written** wins
- More **specific selectors** override general ones
- **!important** overrides everything (use carefully)

### **Example:**

```
p {  
color: blue !important;  
}
```

# CSS Selectors

CSS selectors define **which HTML elements** a style rule applies to. Mastering selectors is essential because they control precision, reusability, and priority in CSS.

---

## 1. Element (Type) Selector

Targets all elements of a specific HTML tag.

```
p {  
  color: blue;  
}  
<p>This paragraph is blue</p>
```

---

## 2. Class Selector

Targets elements with a specific class attribute. Classes are reusable.

```
.highlight {  
  background-color: yellow;  
}  
<p class="highlight">Highlighted text</p>  
<div class="highlight">Highlighted div</div>
```

---

## 3. ID Selector

Targets **one unique element** using the id attribute.

```
#main-title {  
  font-size: 32px;  
}  
<h1 id="main-title">Main Title</h1>
```

### Rules:

- IDs must be unique
- Higher priority than classes

## 4. Universal Selector

Targets **all elements** on the page.

```
* {  
margin: 0;  
padding: 0;  
}
```

---

## 5. Grouping Selector

Applies the same style to multiple selectors.

```
h1, h2, h3 {  
font-family: Arial;  
}
```

---

## 6. Attribute Selectors

Attribute selectors target elements based on the presence or value of an attribute.

---

### 6.1 Attribute Exists Selector

```
input[required] {  
border: 2px solid red;  
}
```

Targets inputs that have the required attribute.

---

### 6.2 Exact Attribute Value Selector

```
input[type="password"] {  
background-color: lightgray;  
}
```

---

### 6.3 Attribute Contains Selector

```
a[href*="google"] {  
color: green;  
}
```

Targets links that contain the word "google" in the URL.

## 6.4 Attribute Starts With

```
a[href^="https"] {  
  color: blue;  
}
```

---

## 6.5 Attribute Ends With

```
a[href$=".pdf"] {  
  color: red;  
}
```

---

# 7. Combinator Selectors

Combinators define relationships between elements.

---

## 7.1 Descendant Selector (space)

Targets elements inside another element.

```
div p {  
  color: purple;  
}
```

---

## 7.2 Child Selector (>)

Targets **direct children only**.

```
div > p {  
  color: orange;  
}
```

---

## 7.3 Adjacent Sibling Selector (+)

Targets the **next immediate sibling**.

```
h1 + p {  
  color: red;  
}
```

## 7.4 General Sibling Selector (~)

Targets all siblings after an element.

```
h1 ~ p {  
  color: gray;  
}
```

## 8. Pseudo-Classes

Pseudo-classes define a **special state** of an element.

Pseudo-class	Applies to	Description
:link	<a>	Unvisited links
:visited	<a>	Visited links
:hover	All	When user hovers with mouse
:active	All	While element is being clicked
:focus	Inputs, buttons	When element is focused
:focus-visible	Inputs	Focus shown only for keyboard navigation
:focus-within	Containers	When any child has focus
:target	All	Element targeted by URL fragment (#id)
:empty	All	Elements with no children or text
:not(selector)	All	Excludes specific selector
:root	html	The root element of the document
:default	Form elements	Default option (radio/checkbox)
:checked	Checkbox, radio	Checked inputs
:disabled	Form elements	Disabled elements
:enabled	Form elements	Enabled elements
:readonly	Inputs	Read-only inputs

:required	Inputs	Required inputs
:optional	Inputs	Inputs without required
:valid	Inputs	Input with valid value
:invalid	Inputs	Input with invalid value
:in-range	Inputs	Value within allowed range
:out-of-range	Inputs	Value outside allowed range
:placeholder-shown	Inputs	Placeholder text is visible
:first-child	All	First child of parent
:last-child	All	Last child of parent
:only-child	All	Element is the only child
:nth-child(n)	All	Matches nth child
:nth-last-child(n)	All	Matches nth child from end
:first-of-type	All	First element of its type
:last-of-type	All	Last element of its type
:only-of-type	All	Only element of its type
:nth-of-type(n)	All	nth element of its type
:nth-last-of-type(n)	All	nth element of its type from end
:is(selector)	All	Matches any selector in list
:where(selector)	All	Like :is() but zero specificity
:has(selector)	Containers	Parent selector (modern CSS)
:lang()	All	Elements with specific language

## 8. Pseudo-Elements

Pseudo-elements style **specific parts** of an element.

---

### 8.1 First Letter

```
p::first-letter {  
font-size: 32px;  
}
```

---

### 8.2 First Line

```
p::first-line {  
font-weight: bold;  
}
```

---

### 8.3 Before & After

```
h1::before {  
content: "💧";  
}  
  
h1::after {  
content: "🚀";  
}
```

---

## 10. Selector Priority (Specificity)

From lowest to highest priority:

1. Element selectors
2. Class selectors
3. ID selectors
4. Inline styles

# Fonts & Text

This section covers **EVERY core font and text-related CSS property**, what it does, its possible values, and a **full visual HTML + CSS example** for each.

## FONT-FAMILY

### What it controls

Which font is used to render text.

### Common values

- Specific fonts: Arial, Times New Roman, Roboto
- Generic families:
  - serif
  - sans-serif
  - monospace
  - cursive
  - fantasy

The screenshot shows a web browser interface with a code editor on the left and a live preview on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.serif { font-family: serif; }
.sans { font-family: sans-serif; }
.mono { font-family: monospace; }
.cursive { font-family: cursive; }
.fantasy { font-family: fantasy; }

.box {
padding: 10px;
margin: 8px;
background: #f0f0f0;
}
</style>
</head>
</html>

<div class="box serif">Serif font</div>
<div class="box sans">Sans-serif font</div>
<div class="box mono">Monospace font</div>
<div class="box cursive">Cursive font</div>
<div class="box fantasy">Fantasy font</div>

</body>
</html>
```

The live preview on the right shows five boxes, each containing text and a label: "Serif font", "Sans-serif font", "Monospace font", "Cursive font", and "Fantasy font". The boxes are styled with a light gray background, 10px padding, and 8px margin. The text in each box is rendered in the corresponding font family.

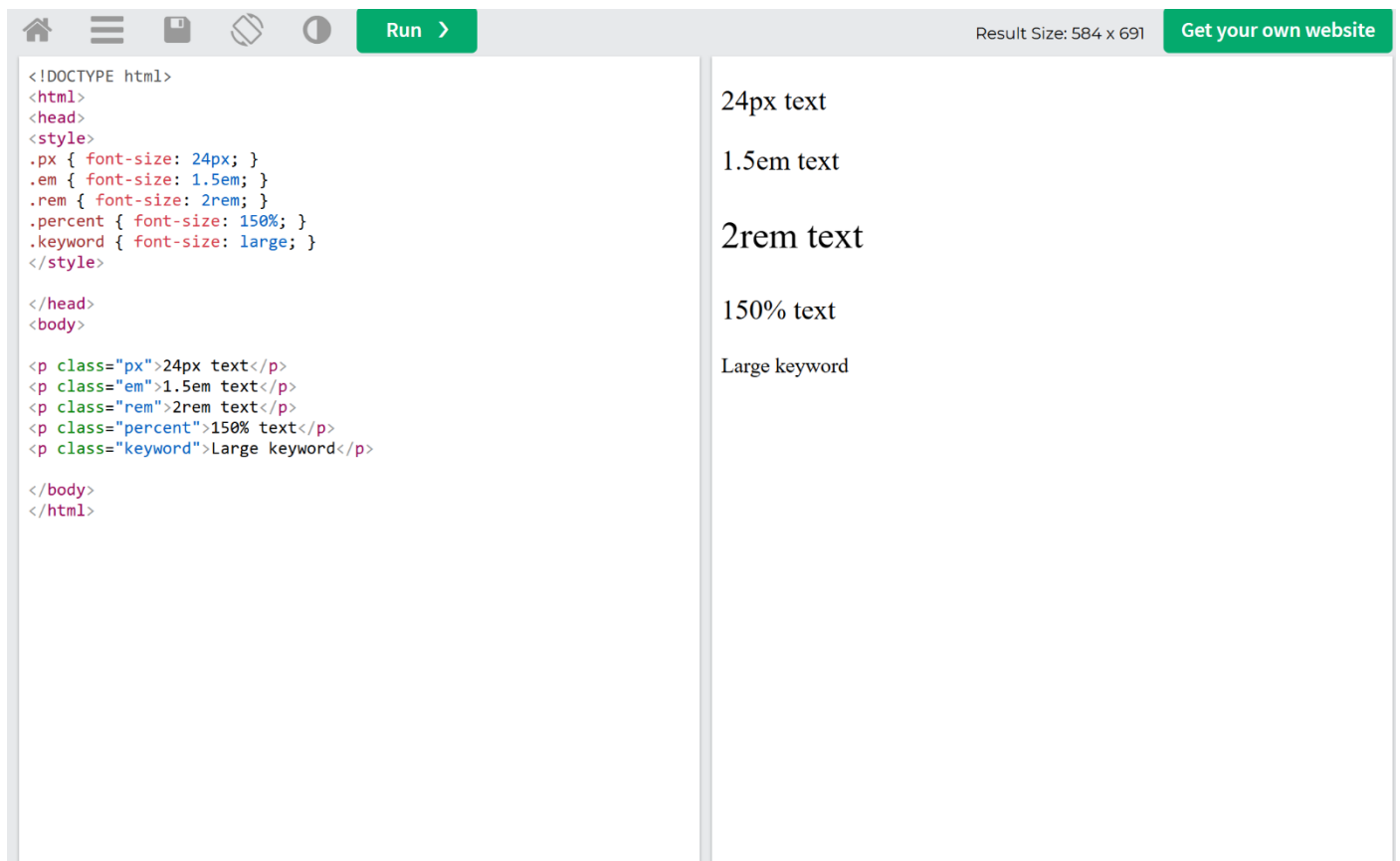
# FONT-SIZE

## What it controls

Size of the text.

## Values

- px
- em
- rem
- %
- Keywords: small, medium, large



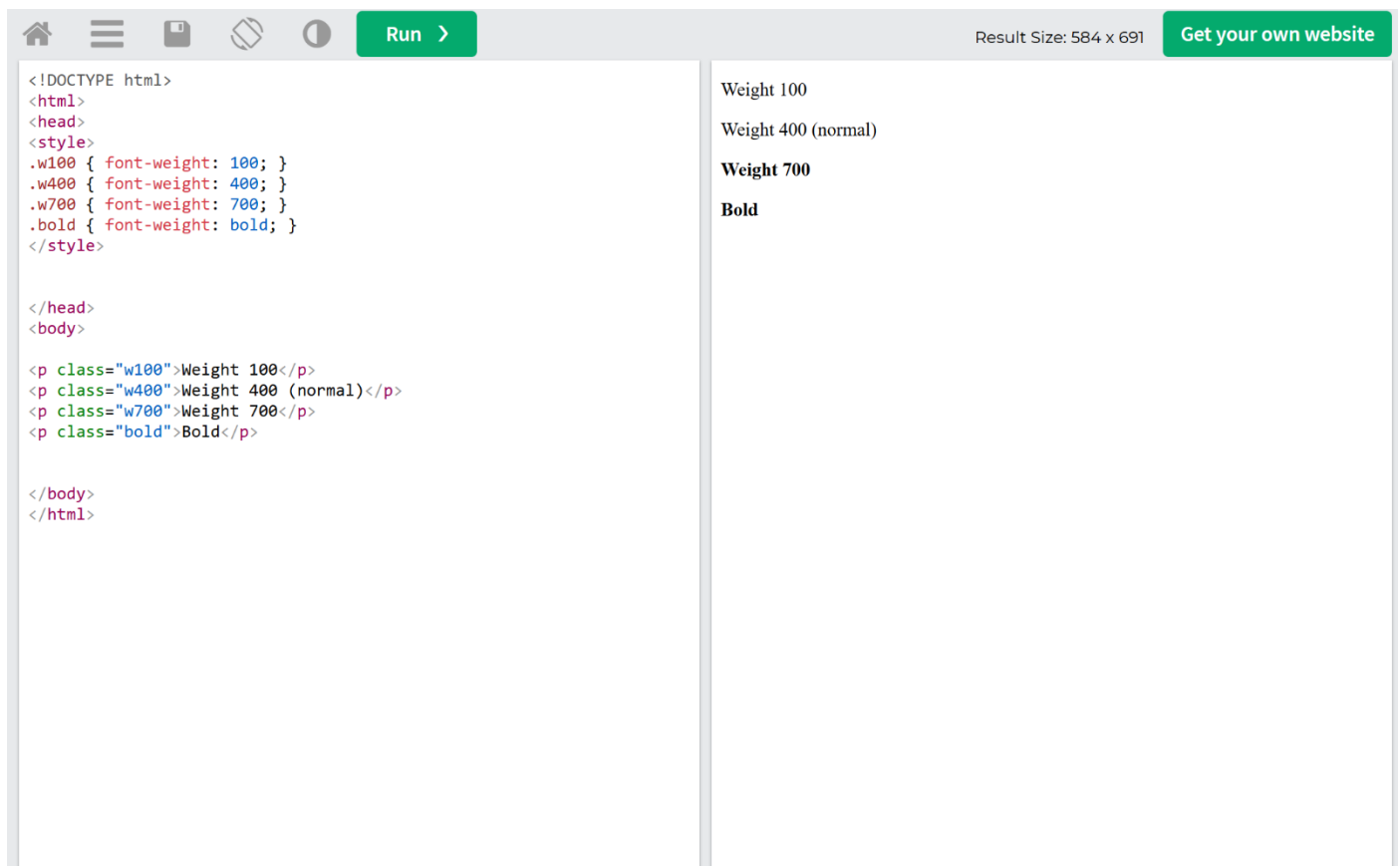
# FONT-WEIGHT

## What it controls

Thickness (boldness) of text.

## Values

- Keywords: normal, bold, lighter, bolder
- Numbers: 100 → 900



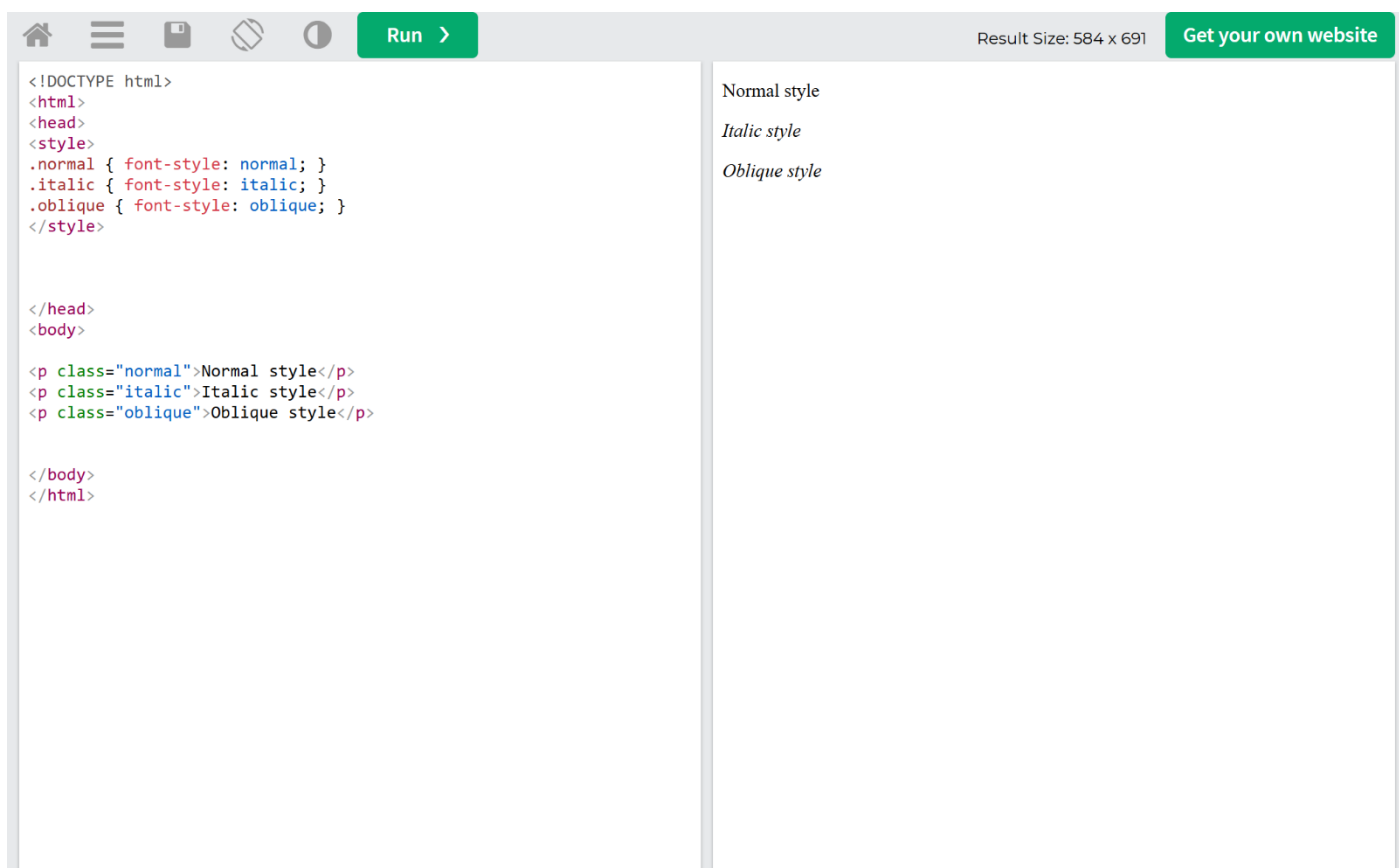
# FONT-STYLE

## What it controls

Italic or normal text.

## Values

- normal
- italic
- oblique



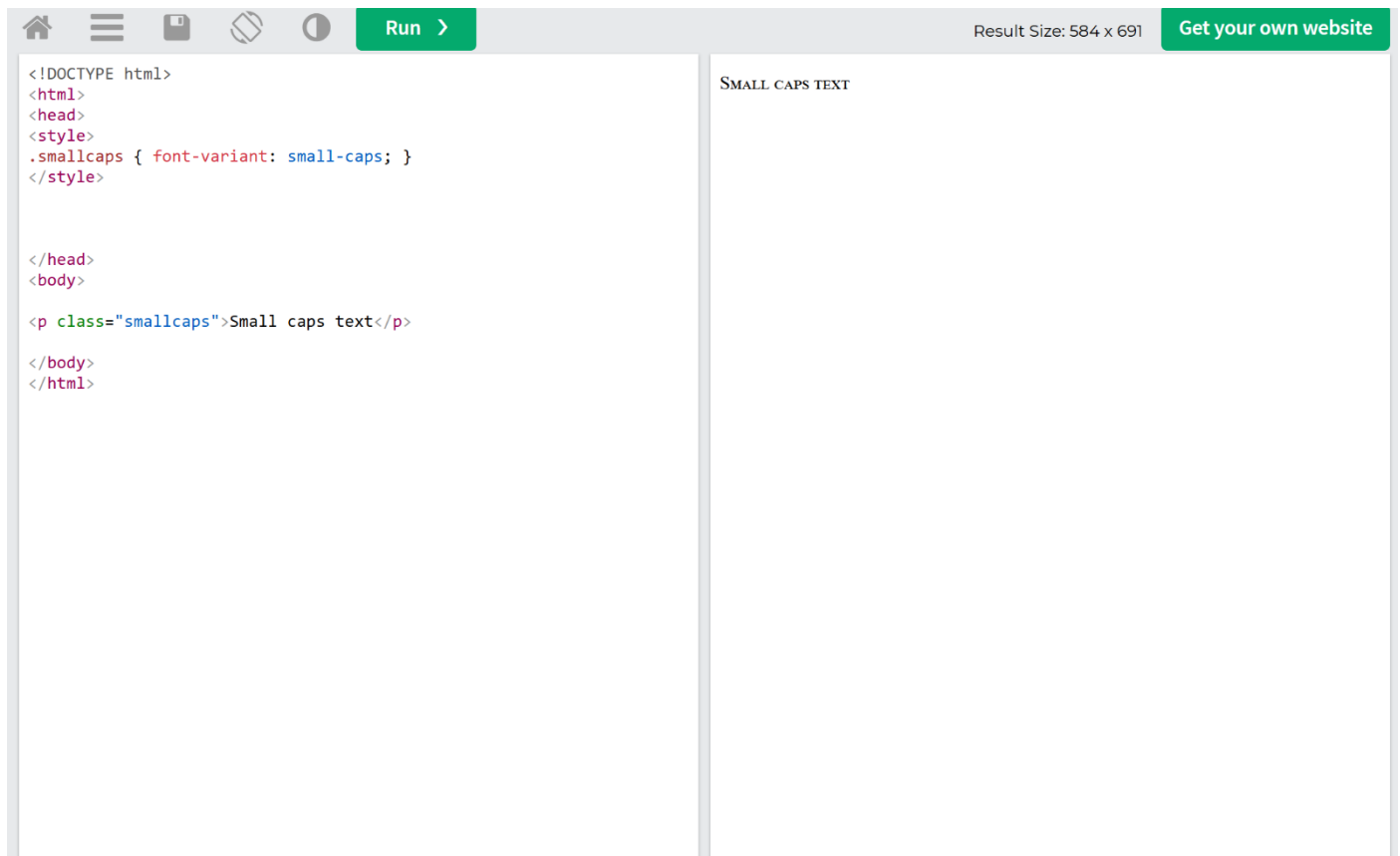
# FONT-VARIANT

## What it controls

Alternative glyphs like small caps.

## Values

- normal
- small-caps



# LINE-HEIGHT

## What it controls

Vertical spacing between lines of text.

## Values

- Unitless number
- px
- em
- normal

The screenshot shows a web browser interface with a code editor on the left and a preview on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.tight { line-height: 1; }
.normal { line-height: 1.6; }
.loose { line-height: 2.5; }

.box {
width: 300px;
background: #eef;
margin: 10px;
}
</style>

</head>
<body>

<div class="box tight">Tight line height<br>Second line</div>
<div class="box normal">Normal line height<br>Second line</div>
<div class="box loose">Loose line height<br>Second line</div>

</body>
</html>
```

The preview on the right shows three examples of text with different line heights, each in a light blue box:

- Tight line height:** The text "Tight line height" and "Second line" are very close together.
- Normal line height:** The text "Normal line height" and "Second line" are spaced normally.
- Loose line height:** The text "Loose line height" and "Second line" are spaced very far apart.

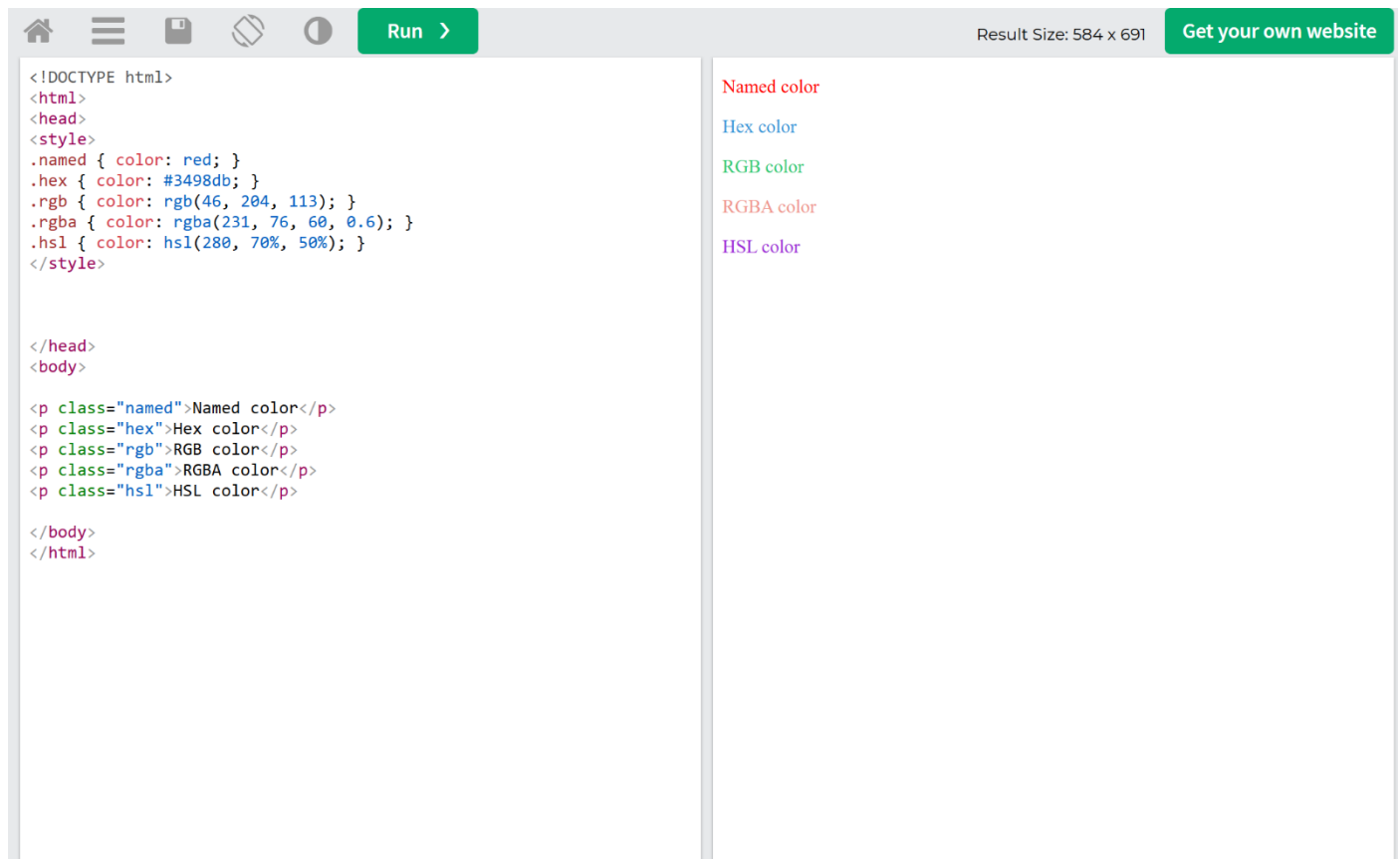
# COLOR

## What it controls

Text color.

## Values

- Named colors
- hex
- rgb()
- rgba()
- hsl() (Hue, saturation, lightness)
- hsla() (Hue, saturation, lightness, alpha/opacity)



The screenshot shows a web browser interface with a code editor on the left and a preview on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.named { color: red; }
.hex { color: #3498db; }
.rgb { color: rgb(46, 204, 113); }
.rgba { color: rgba(231, 76, 60, 0.6); }
.hsl { color: hsl(280, 70%, 50%); }
</style>

</head>
<body>

<p class="named">Named color</p>
<p class="hex">Hex color</p>
<p class="rgb">RGB color</p>
<p class="rgba">RGBA color</p>
<p class="hsl">HSL color</p>

</body>
</html>
```

The preview on the right shows the rendered output of the code, with each color name displayed in its corresponding color:

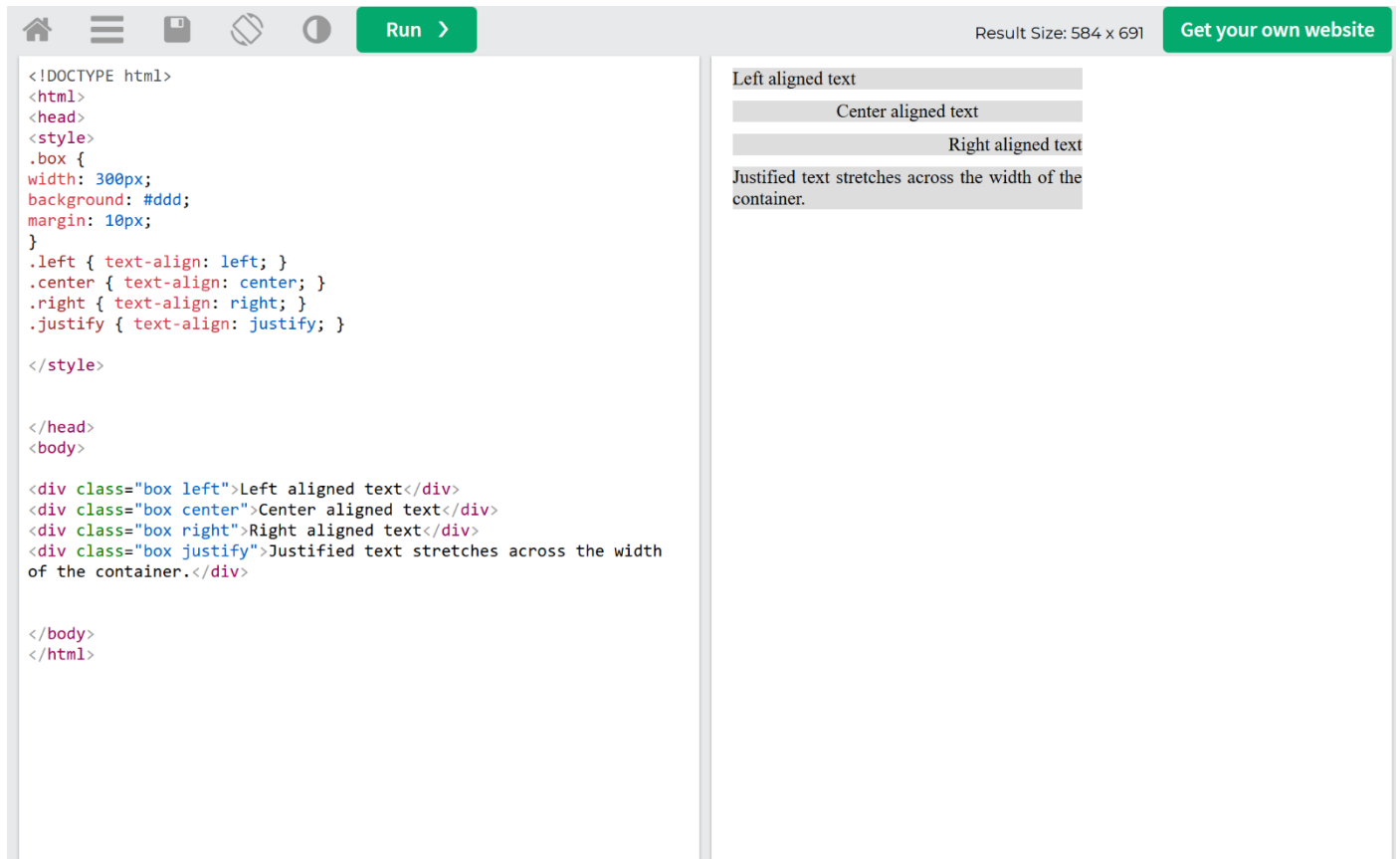
- Named color (red)
- Hex color (blue)
- RGB color (green)
- RGBA color (orange)
- HSL color (purple)

The browser interface includes a top navigation bar with a home icon, a menu icon, a save icon, a refresh icon, and a 'Run' button. The result size is indicated as 584 x 691. A green button in the top right corner says 'Get your own website'.

# TEXT-ALIGN

## Values

- left
- right
- center
- justify



The screenshot shows a web browser interface with a code editor on the left and a preview on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
width: 300px;
background: #ddd;
margin: 10px;
}
.left { text-align: left; }
.center { text-align: center; }
.right { text-align: right; }
.justify { text-align: justify; }
</style>

</head>
<body>

<div class="box left">Left aligned text</div>
<div class="box center">Center aligned text</div>
<div class="box right">Right aligned text</div>
<div class="box justify">Justified text stretches across the width
of the container.</div>

</body>
</html>
```

The preview on the right shows the rendered output: four horizontal bars with different text alignments. The first bar is left-aligned, the second is center-aligned, the third is right-aligned, and the fourth is justified, stretching across the width of the container.

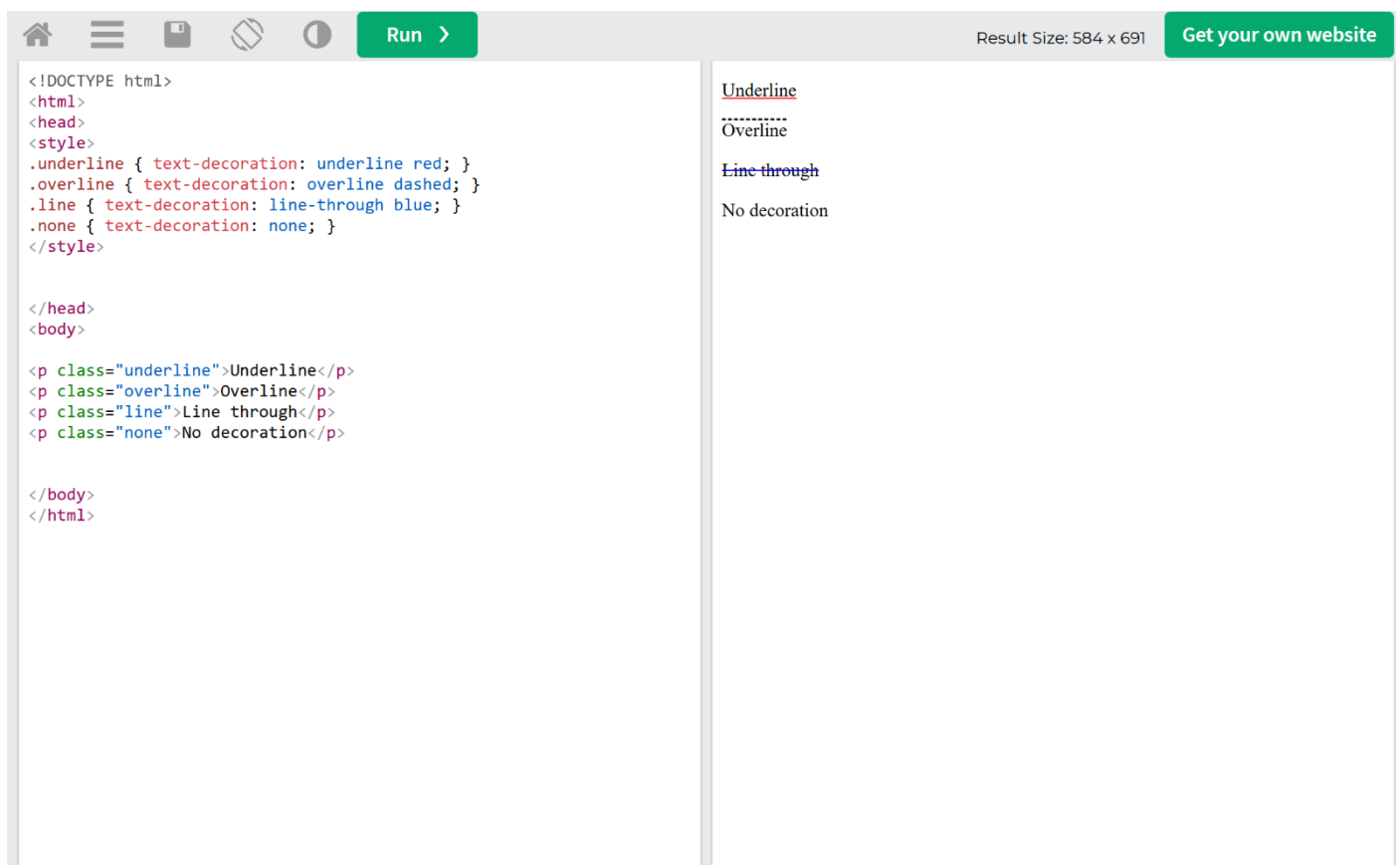
# TEXT-DECORATION

## What it controls

Lines on text.

## Values

- underline
- overline
- line-through
- none



The screenshot shows a web development tool interface. On the left, there is a code editor with the following HTML and CSS code:

```
<!DOCTYPE html>
<html>
<head>
<style>
.underline { text-decoration: underline red; }
.overline { text-decoration: overline dashed; }
.line { text-decoration: line-through blue; }
.none { text-decoration: none; }
</style>

</head>
<body>

<p class="underline">Underline</p>
<p class="overline">Overline</p>
<p class="line">Line through</p>
<p class="none">No decoration</p>

</body>
</html>
```

On the right, there is a preview window showing the rendered output:

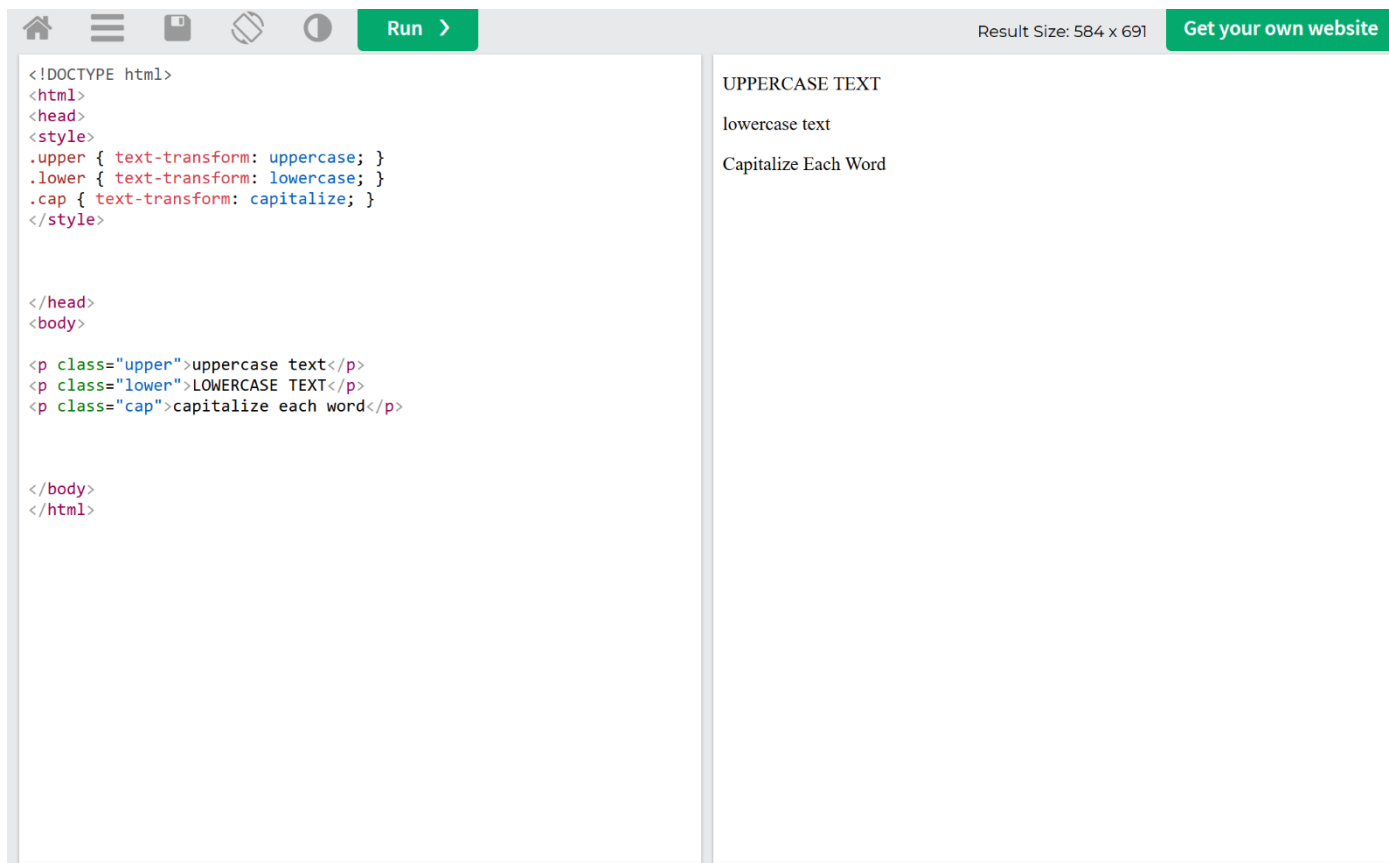
Underline  
-----  
Overline  
~~Line through~~  
No decoration

The preview window also displays the text "Result Size: 584 x 691" and a button that says "Get your own website".

## TEXT-TRANSFORM

### Values

- uppercase
- lowercase
- capitalize



The screenshot shows a web browser interface with a code editor on the left and a preview area on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.upper { text-transform: uppercase; }
.lower { text-transform: lowercase; }
.cap { text-transform: capitalize; }
</style>

</head>
<body>

<p class="upper">uppercase text</p>
<p class="lower">LOWERCASE TEXT</p>
<p class="cap">capitalize each word</p>

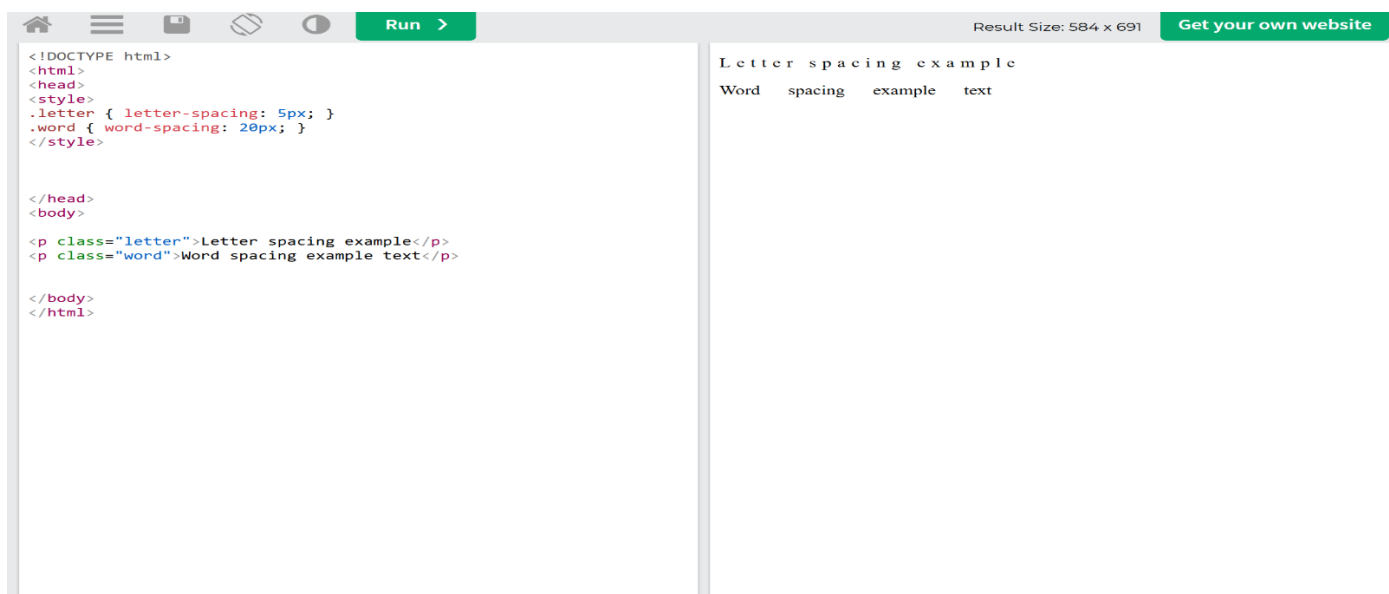
</body>
</html>
```

The preview area displays the rendered output:

UPPERCASE TEXT  
lowercase text  
Capitalize Each Word

The browser interface includes a top navigation bar with a 'Run' button and a 'Get your own website' button. The result size is indicated as 584 x 691.

## LETTER-SPACING & WORD-SPACING



The screenshot shows a web browser interface with a code editor on the left and a preview area on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.letter { letter-spacing: 5px; }
.word { word-spacing: 20px; }
</style>

</head>
<body>

<p class="letter">Letter spacing example</p>
<p class="word">Word spacing example text</p>

</body>
</html>
```

The preview area displays the rendered output:

Letter spacing example  
Word spacing example text

The browser interface includes a top navigation bar with a 'Run' button and a 'Get your own website' button. The result size is indicated as 584 x 691.

# WHITE-SPACE

## Values

- normal
- nowrap
- pre
- pre-line
- pre-wrap

The screenshot shows a web browser interface with a code editor on the left and a preview on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
width: 200px;
background: #eee;
margin: 10px;
}

.nowrap { white-space: nowrap; }
.pre { white-space: pre; }
.prewrap { white-space: pre-wrap; }
</style>

</head>
<body>

<div class="box nowrap">This text will not wrap to the next
line</div>
<div class="box pre">Line 1
Line 2
Indent preserved</div>
<div class="box prewrap">Line 1
Line 2
Wraps and preserves spaces</div>

</body>
</html>
```

The preview on the right shows the rendered output of the code. It features a green header with the text "Get your own website" and "Result Size: 584 x 691". The main content area displays three examples of white-space handling:

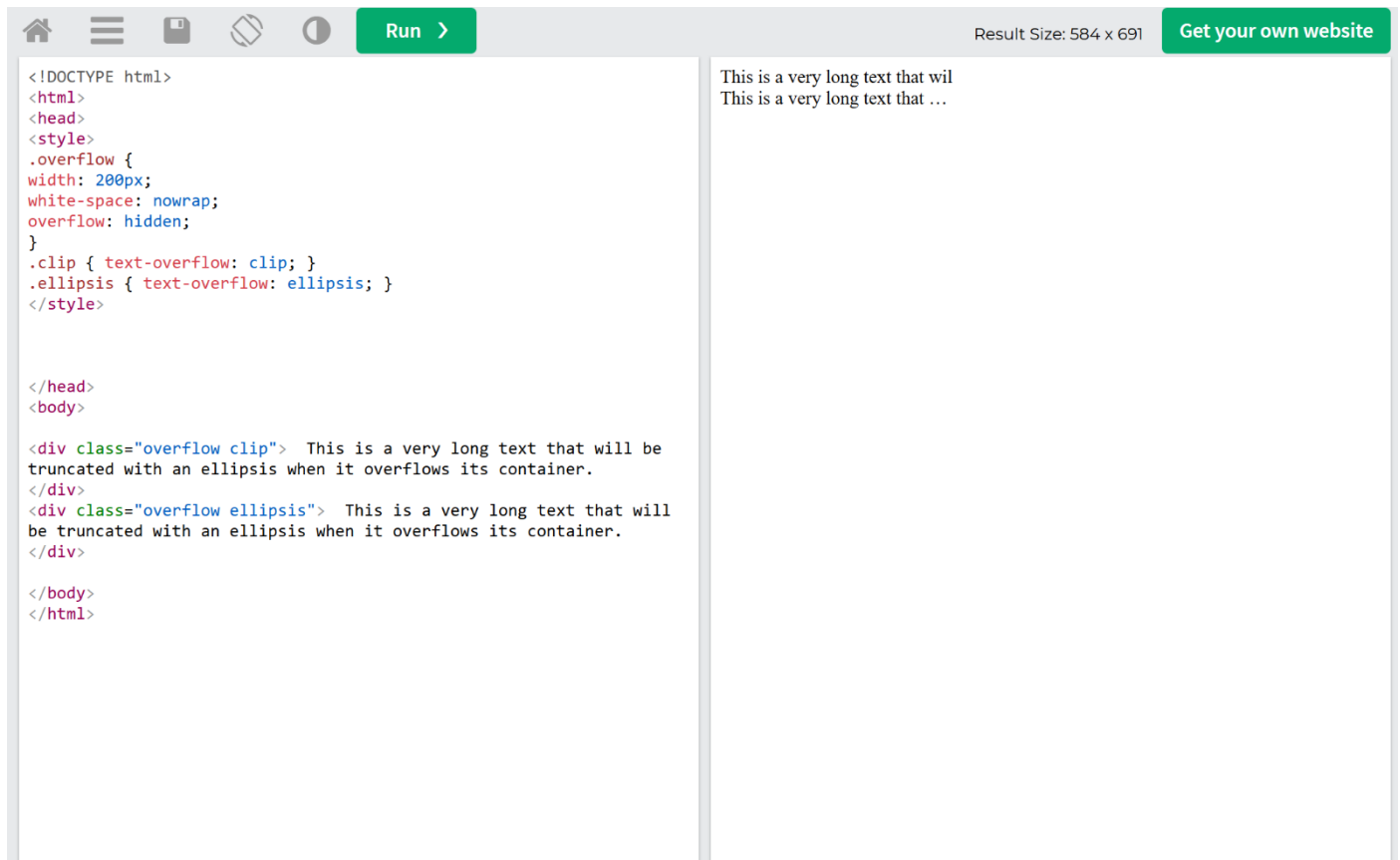
- nowrap:** "This text will not wrap to the next line" (The text is truncated in the image).
- pre:** "Line 1", "Line 2", "Indent preserved" (The text is displayed on separate lines with preserved indentation).
- pre-wrap:** "Line 1", "Line 2", "Wraps and preserves spaces" (The text is displayed on separate lines with preserved indentation and wrapping).

## TEXT-OVERFLOW

**Text overflow** refers to how content is handled when it is too large to fit within its container box.

### Values

- clip
- ellipsis



The screenshot shows a web development tool interface. On the left is a code editor with the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.overflow {
width: 200px;
white-space: nowrap;
overflow: hidden;
}
.clip { text-overflow: clip; }
.ellipsis { text-overflow: ellipsis; }
</style>

</head>
<body>

<div class="overflow clip"> This is a very long text that will be
truncated with an ellipsis when it overflows its container.
</div>
<div class="overflow ellipsis"> This is a very long text that will
be truncated with an ellipsis when it overflows its container.
</div>

</body>
</html>
```

On the right is a preview window showing the rendered output:

```
This is a very long text that wil
This is a very long text that ...
```

The preview window shows two lines of text. The first line is truncated with an ellipsis, and the second line is truncated with an ellipsis. The text is displayed in a monospace font.

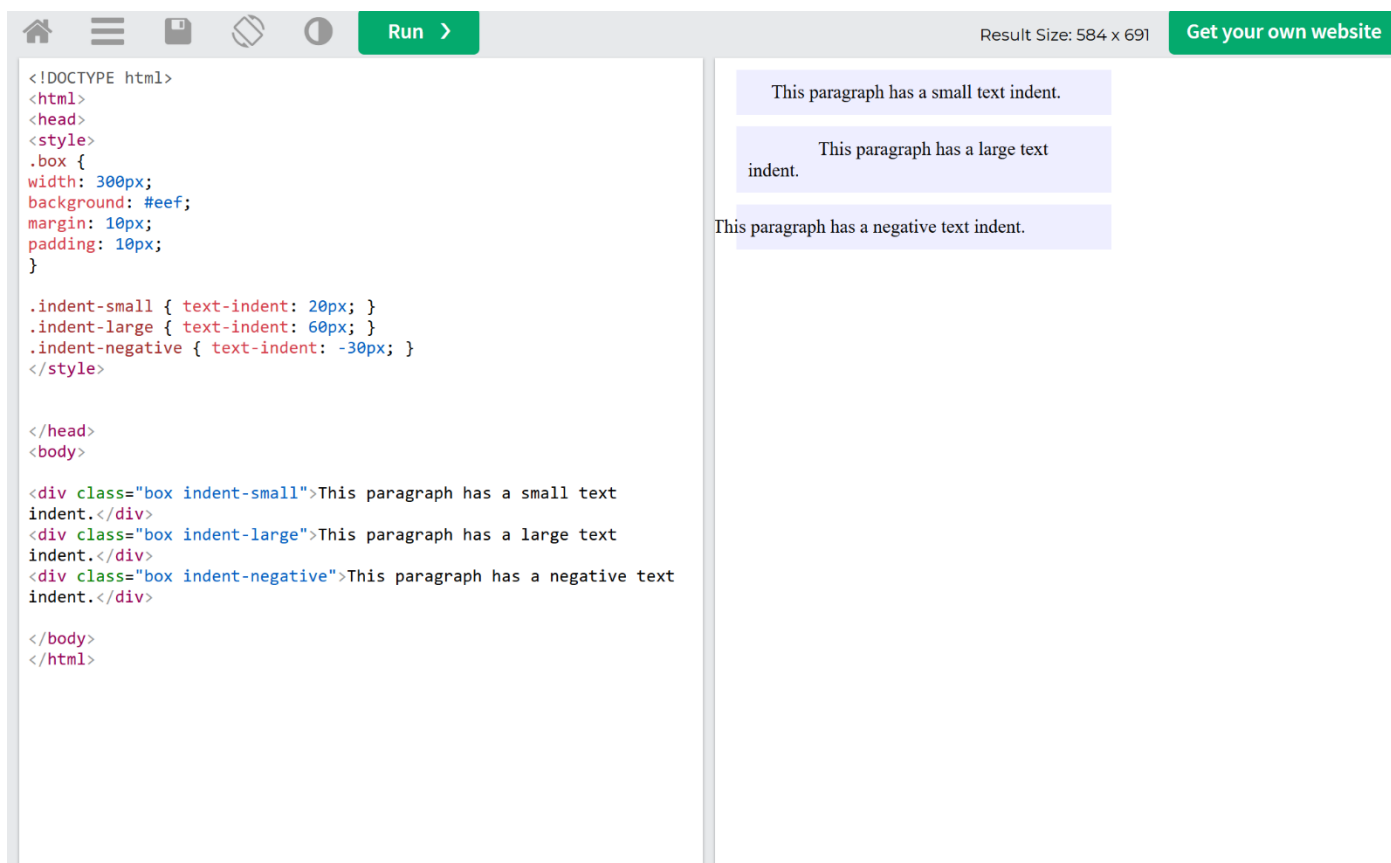
# TEXT-INDENT

## What it controls

Moves the **first line** of text inward from the left (or right in RTL).

## Common values

- px, em, rem, %
- Negative values allowed



The screenshot shows a web browser interface with a code editor on the left and a rendered preview on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
width: 300px;
background: #eef;
margin: 10px;
padding: 10px;
}

.indent-small { text-indent: 20px; }
.indent-large { text-indent: 60px; }
.indent-negative { text-indent: -30px; }
</style>

</head>
<body>

<div class="box indent-small">This paragraph has a small text
indent.</div>
<div class="box indent-large">This paragraph has a large text
indent.</div>
<div class="box indent-negative">This paragraph has a negative text
indent.</div>

</body>
</html>
```

The rendered preview shows three paragraphs of text, each enclosed in a light blue box. The first paragraph is indented 20px, the second is indented 60px, and the third is indented -30px (negative indent).

# TEXT-SHADOW

## What it controls

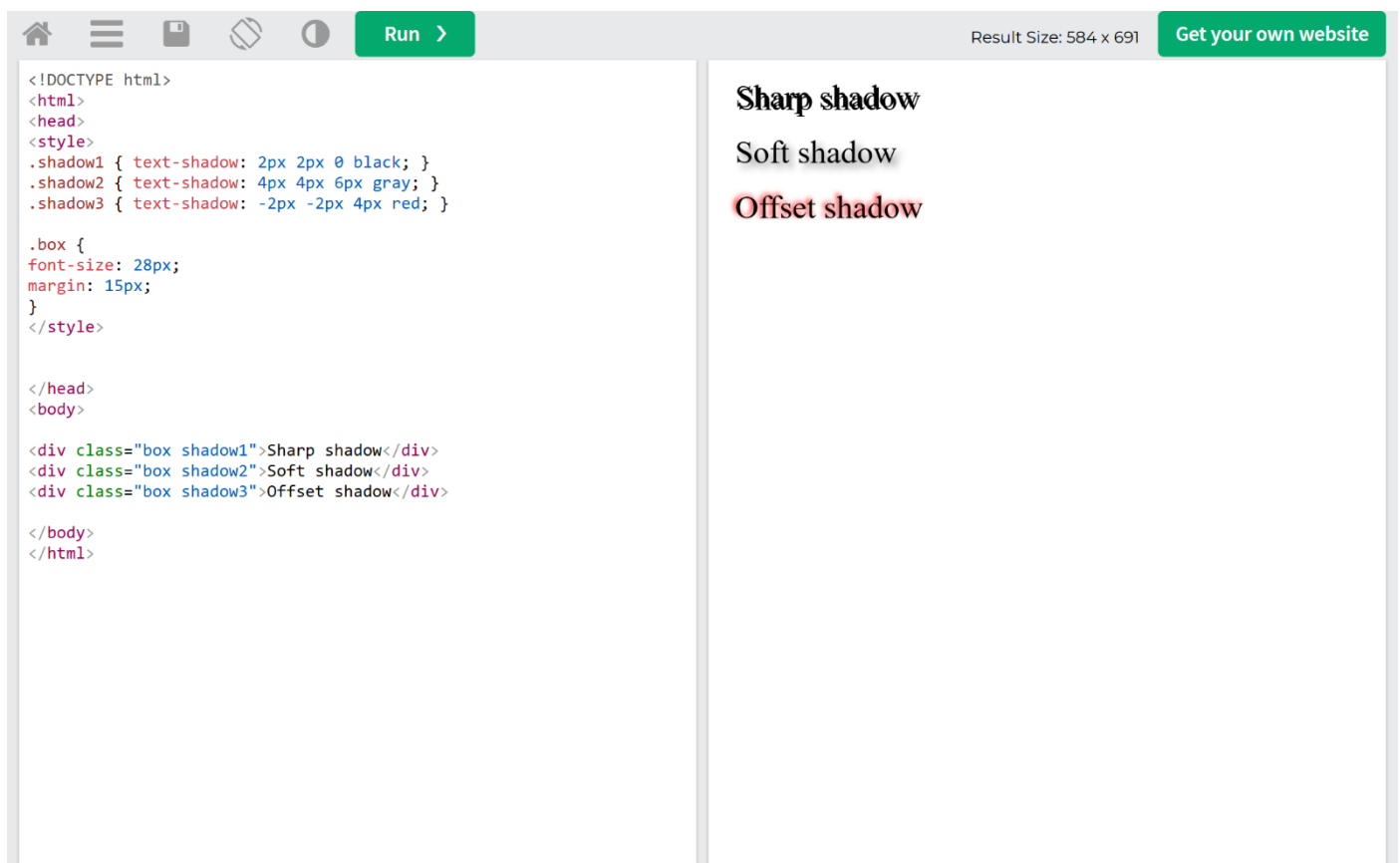
Adds shadow effects to text.

## Syntax

text-shadow: x-offset y-offset blur color;

## Values

- Horizontal offset
- Vertical offset
- Blur radius (optional)
- Color



The screenshot shows a web browser interface with a code editor on the left and a rendered preview on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.shadow1 { text-shadow: 2px 2px 0 black; }
.shadow2 { text-shadow: 4px 4px 6px gray; }
.shadow3 { text-shadow: -2px -2px 4px red; }

.box {
font-size: 28px;
margin: 15px;
}
</style>
</head>
<body>

<div class="box shadow1">Sharp shadow</div>
<div class="box shadow2">Soft shadow</div>
<div class="box shadow3">Offset shadow</div>

</body>
</html>
```

The rendered preview on the right shows three lines of text, each with a different shadow effect:

- Sharp shadow**: Text with a sharp black shadow.
- Soft shadow**: Text with a soft, gray shadow.
- Offset shadow**: Text with a red shadow offset to the top-left.

The browser interface includes a top bar with navigation icons, a 'Run' button, and a 'Get your own website' button. The result size is indicated as 584 x 691.

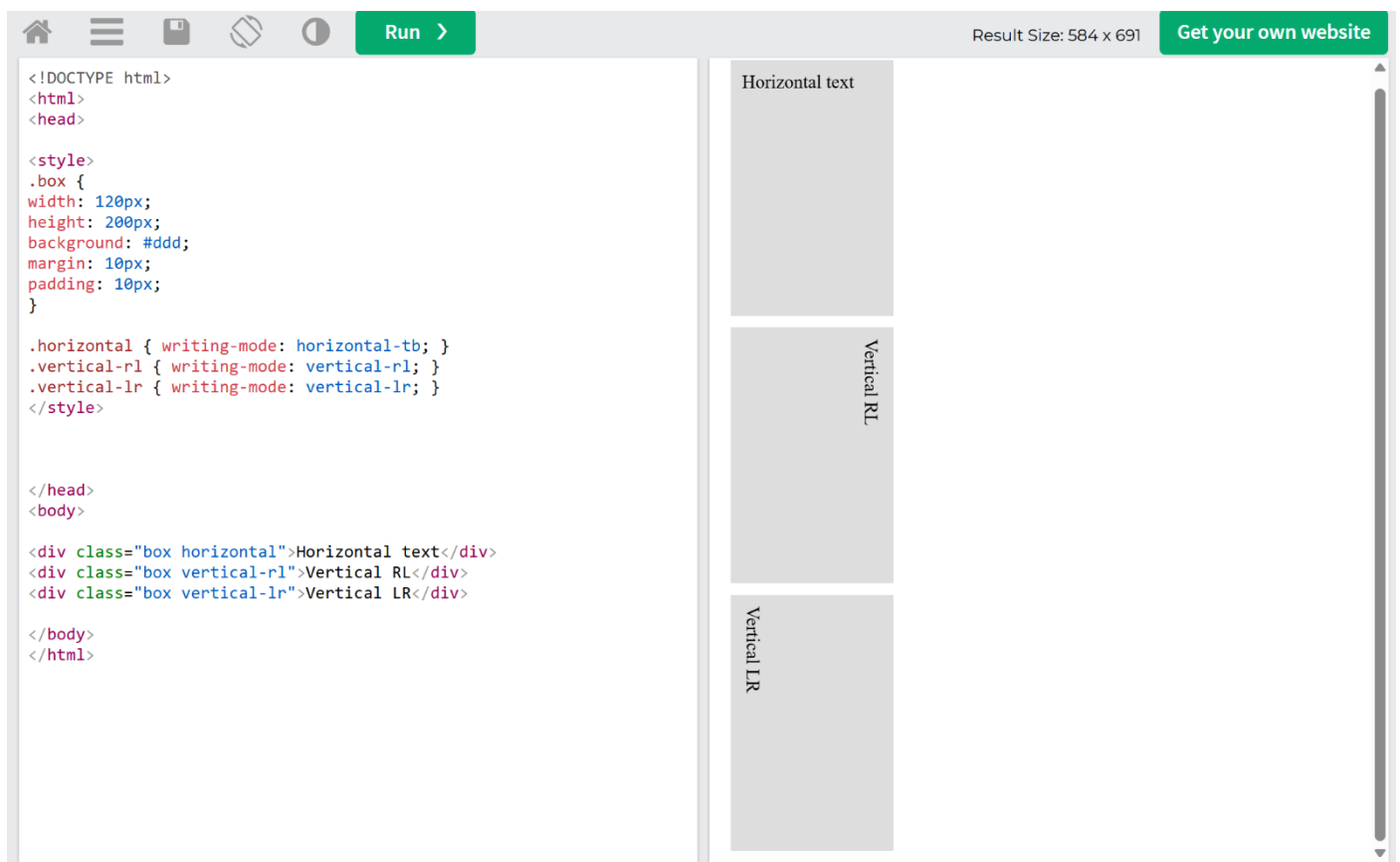
# WRITING-MODE

## What it controls

Sets **text flow direction** (horizontal or vertical).

## Common values

- horizontal-tb (default)
- vertical-rl
- vertical-lr



# TEXT-ORIENTATION

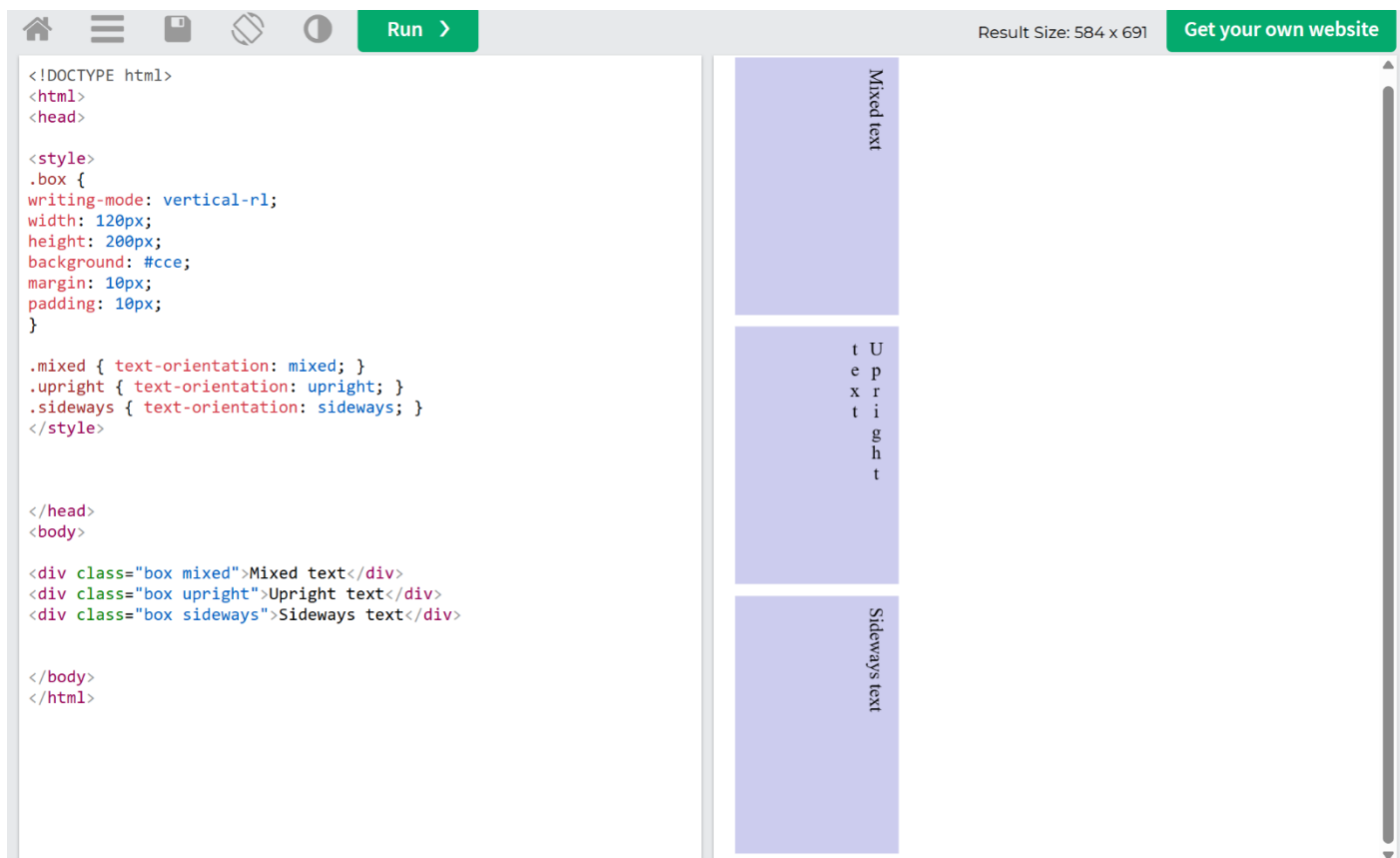
## What it controls

How characters are oriented **inside vertical writing modes**.

⚠ Has effect **only when writing-mode is vertical**.

## Values

- mixed (default)
- upright
- sideways



## Web Fonts

### WHAT ARE WEB FONTS?

By default, browsers can only use **fonts installed on the user's system** (Arial, Times, etc.).

**Web fonts** allow you to:

- Use **any custom font**
- Ensure **consistent design** across devices
- Control typography precisely

They are downloaded by the browser just like images or CSS.

---

### *@FONT-FACE*

#### What it does

Defines a **custom font** and gives it a name that CSS can use.

Once declared, the font behaves like any system font.

---

### *BASIC @FONT-FACE SYNTAX*

```
@font-face {  
font-family: "MyFont";  
src: url("MyFont.woff2");  
}
```

- font-family → name you choose
- src → font file location

## FONT FORMATS (IMPORTANT)

Common formats:

- woff2  best (smallest, fastest)
- woff
- ttf
- otf

Browsers try sources **top to bottom**.

---

## MULTIPLE FONT WEIGHTS

Each weight **must be declared separately**.

```
@font-face {  
font-family: "MyFont";  
src: url("MyFont-Regular.woff2");  
font-weight: 400;  
}
```

```
@font-face {  
font-family: "MyFont";  
src: url("MyFont-Bold.woff2");  
font-weight: 700;  
}  
p { font-family: "MyFont"; }  
strong { font-weight: 700; }
```

## MULTIPLE STYLES (ITALIC)

```
@font-face {  
font-family: "MyFont";  
src: url("MyFont-Italic.woff2");  
font-style: italic;  
}
```

Browser automatically selects the correct file.

## ***FONT-DISPLAY (VERY IMPORTANT)***

Controls **what happens while the font is loading**.

### **Values**

- swap  recommended
- block
- fallback
- optional

```
@font-face {  
font-family: "MyFont";  
src: url("MyFont.woff2");  
font-display: swap;  
}
```

 swap shows fallback text first, then swaps when font loads.

---

## ***USING GOOGLE FONTS***

### ***Option 1 — <link> (most common)***

```
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap"  
rel="stylesheet">  
body {  
font-family: "Roboto", sans-serif;  
}
```

### ***Option 2 — @import (not recommended -> slower than link)***

```
@import url('https://fonts.googleapis.com/css2?family=Roboto&display=swap');
```

# CSS Box Model

The **CSS Box Model** describes how every HTML element is structured and how space is calculated around it. Every element is treated as a rectangular box made up of **four layers**, from inside to outside:

Content → Padding → Border → Margin

Understanding the box model is **critical** for layout, spacing, alignment, and responsive design.

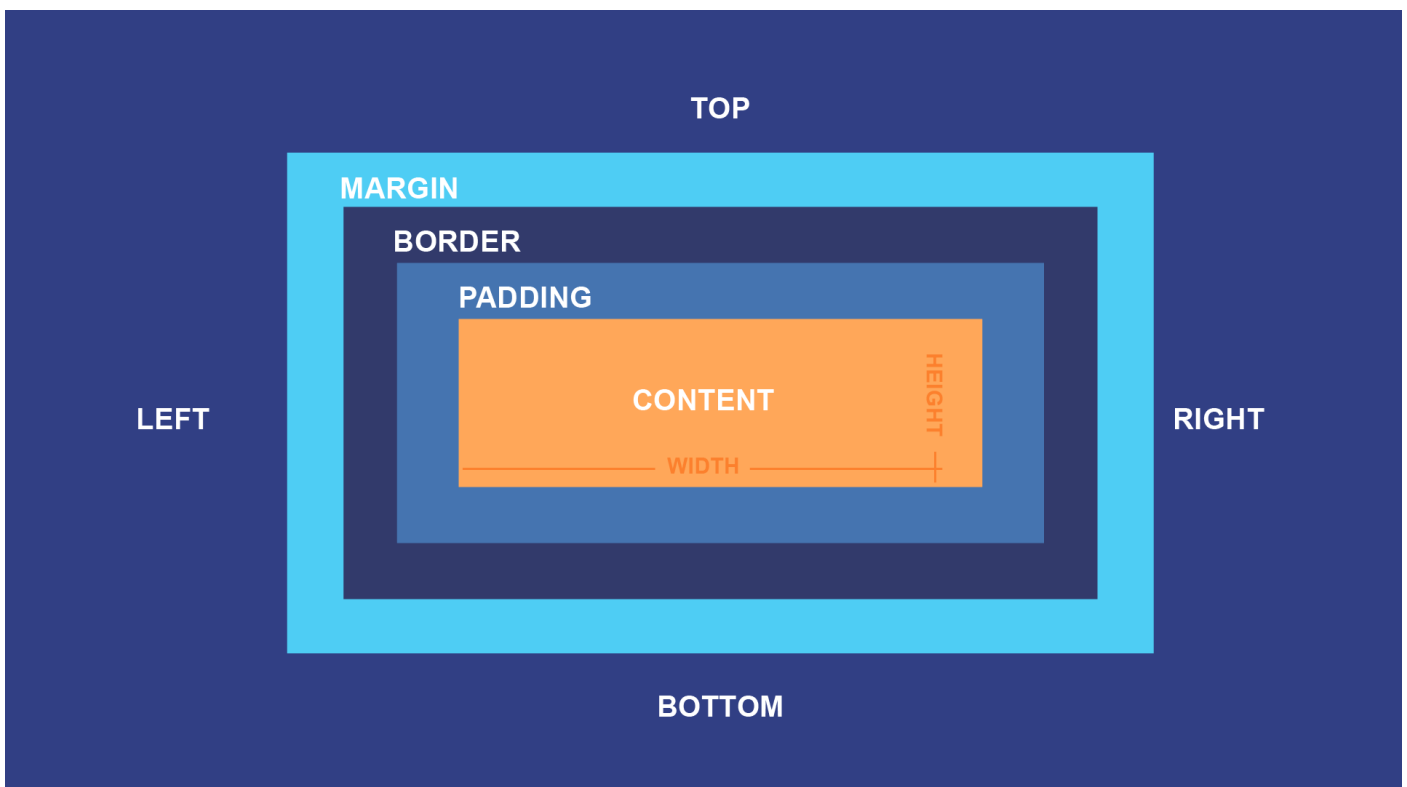
---

## Content

The **content area** holds the actual content of the element such as text, images, or other elements.

Properties that affect content size:

- width
- height
- min-width, max-width
- min-height, max-height



# Padding

Padding is the **space between the content and the border**. It increases the internal spacing of an element.

## Padding properties:

- padding-top
- padding-right
- padding-bottom
- padding-left
- padding (shorthand)

## Padding values:


- px, em, rem, %

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
  width: 200px;
  border: 2px solid black;
  background: lightblue;
  margin-bottom: 20px;
}

.pad-1 { padding: 20px; }
.pad-2 { padding: 10px 30px; }
.pad-3 { padding: 10px 20px 30px; }
.pad-4 { padding: 5px 10px 15px 20px; }
</style>
</head>
<body>

<div class="box pad-1">padding: 20px</div>
<div class="box pad-2">padding: 10px 30px</div>
<div class="box pad-3">padding: 10px 20px 30px</div>
<div class="box pad-4">padding: 5px 10px 15px 20px</div>

</body>
</html>
|
```



How values are applied:

- 1 value → all sides
- 2 values → top/bottom | left/right
- 3 values → top | left/right | bottom
- 4 values → top | right | bottom | left

# Border

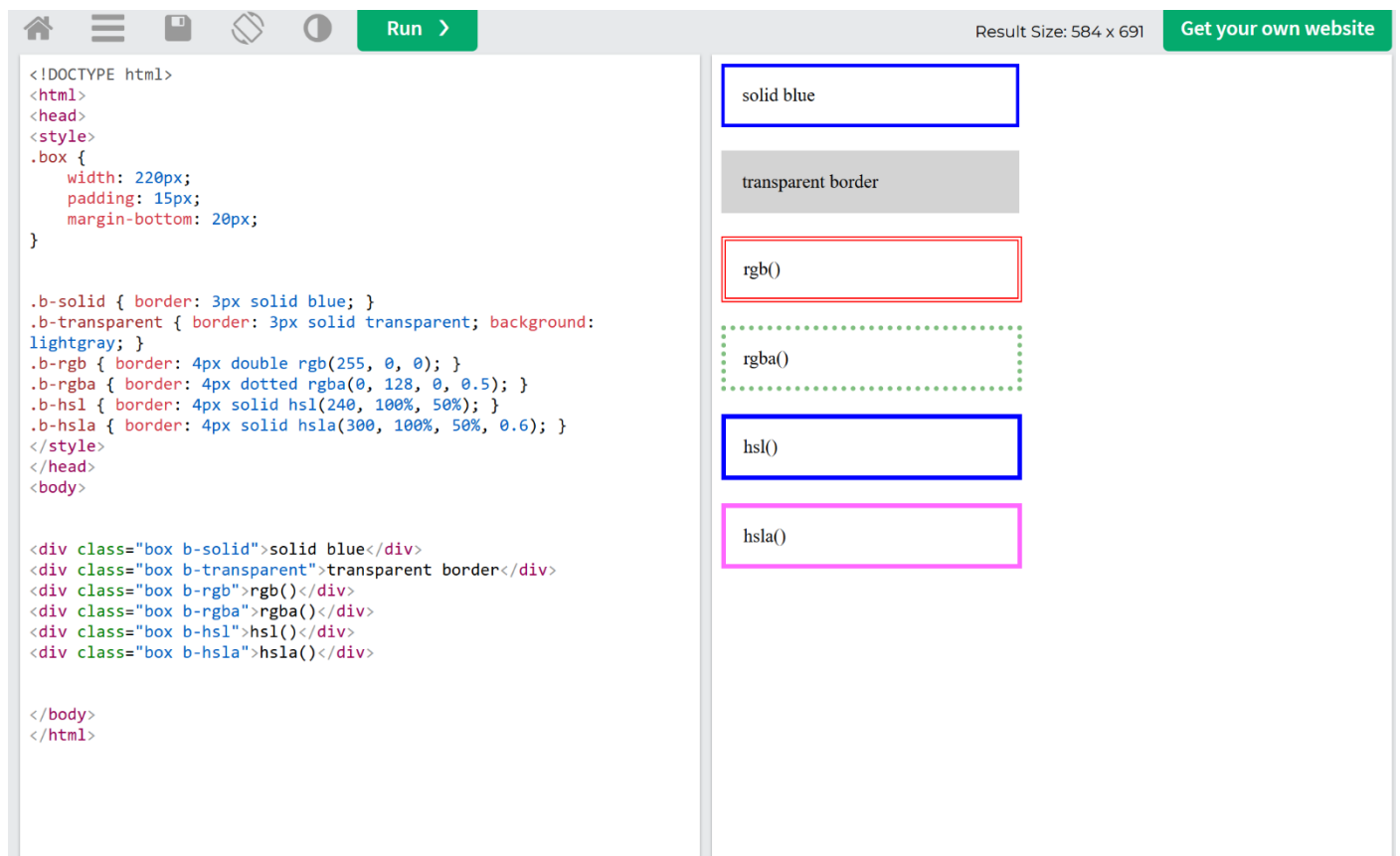
The border wraps around the padding and content.

## Border properties:

- border-width
- border-style
- border-color
- border (shorthand)

## Common border styles:

- solid
- dashed
- dotted
- double
- none



The screenshot shows a web browser interface with a code editor on the left and a preview on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
width: 220px;
padding: 15px;
margin-bottom: 20px;
}

.b-solid { border: 3px solid blue; }
.b-transparent { border: 3px solid transparent; background:
lightgray; }
.b-rgb { border: 4px double rgb(255, 0, 0); }
.b-rgba { border: 4px dotted rgba(0, 128, 0, 0.5); }
.b-hsl { border: 4px solid hsl(240, 100%, 50%); }
.b-hsla { border: 4px solid hsla(300, 100%, 50%, 0.6); }
</style>
</head>
<body>

<div class="box b-solid">solid blue</div>
<div class="box b-transparent">transparent border</div>
<div class="box b-rgb">rgb(</div>
<div class="box b-rgba">rgba(</div>
<div class="box b-hsl">hsl(</div>
<div class="box b-hsla">hsla(</div>

</body>
</html>
```

The preview on the right shows six examples of boxes with different border styles and colors:

- solid blue
- transparent border
- rgb()
- rgba()
- hsl()
- hsla()

## Border Radius

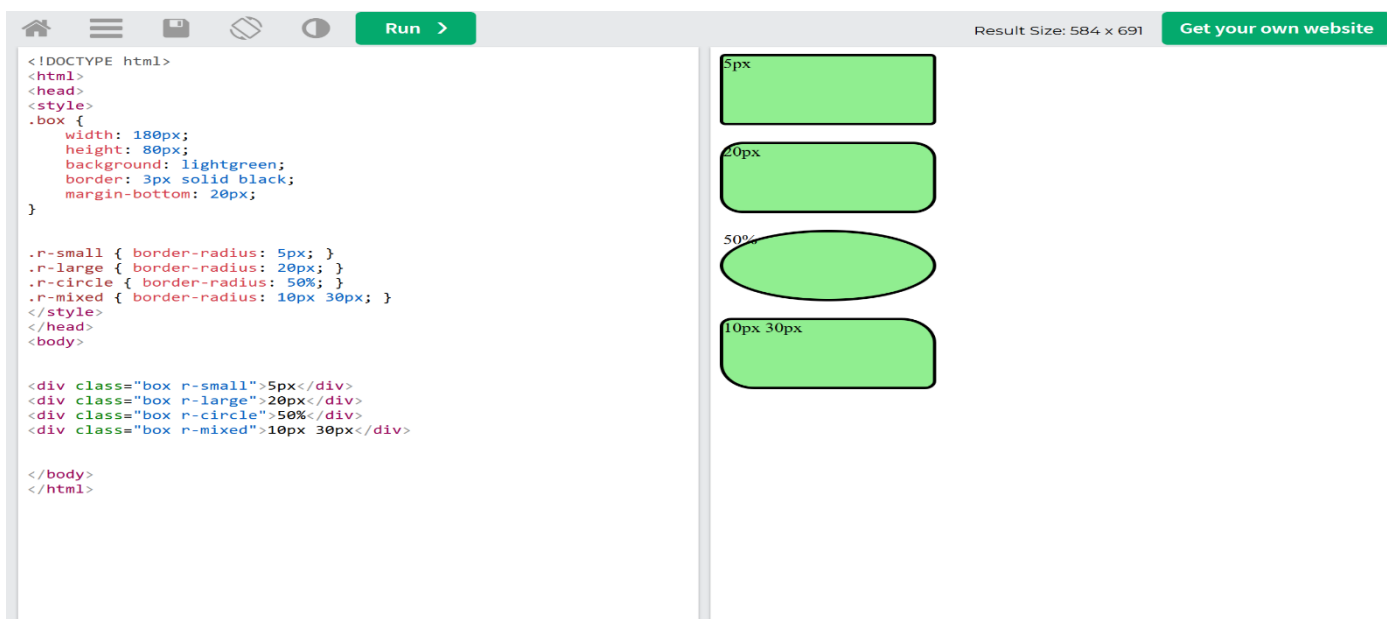
border-radius controls **how rounded the corners** of an element are. It can create anything from slightly rounded rectangles to perfect circles.

### Properties & syntax:

- border-radius
- border-top-left-radius
- border-top-right-radius
- border-bottom-right-radius
- border-bottom-left-radius

### Possible values:

- Lengths: px, em, rem
- Percentages: %
- Multiple values (1–4)



How values work:

- 1 value → all corners
- 2 values → top-left & bottom-right | top-right & bottom-left
- 3 values → top-left | top-right & bottom-right | bottom-left
- 4 values → top-left | top-right | bottom-right | bottom-left

## Border Image

border-image allows you to **use an image or gradient as the border** instead of a solid color.

---

### Required setup:

- A visible border width
  - border-style (or transparent border)
- 

### Main Properties

- **border-image-source:** Specifies the path (URL) to the image you want to use as the border.
  - **border-image-slice:** Tells the browser how to cut the image into nine regions (corners, edges, and center) so it can be stretched or repeated.
  - **border-image-width:** Defines how thick the image border should be, which can be different from the standard border-width.
  - **border-image-outset:** Determines how far the border image area extends beyond the element's actual border box.
  - **border-image-repeat:** Controls whether the image slices should be stretched to fit the area, repeated (tiled), or rounded to fill the space.
  - **border-image (shorthand):** A single property that allows you to set all the above values at once in one line.
- 

### Common values:

- Images (url())
  - Gradients (linear-gradient, radial-gradient)
  - Slice numbers or percentages
  - Repeat options: stretch, repeat, round, space
- 

### Important notes:

- border-image replaces border-color
  - Border radius may not work perfectly with images
  - Mostly used for decorative UI elements
-

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
width: 260px;
padding: 20px;
border: 10px solid transparent;
border-image: linear-gradient(45deg, red, blue, purple) 1;
}
</style>
</head>
<body>

<div class="box">Gradient Border Image</div>

</body>
</html>
```



# Margin

Margin is the **outermost layer**. It creates space **outside** the element, separating it from other elements.

---

## Margin properties:

- margin-top
  - margin-right
  - margin-bottom
  - margin-left
  - margin (shorthand)
- 

## Margin values:

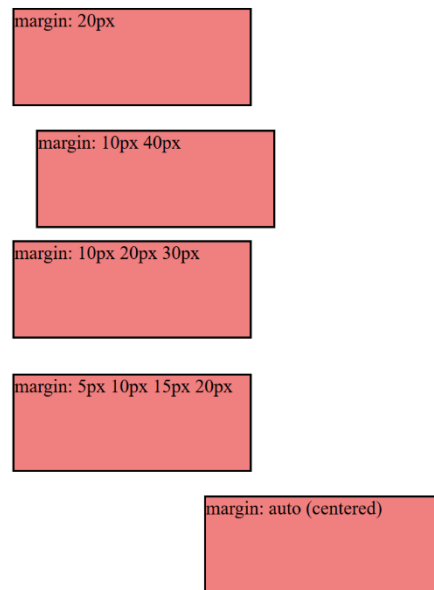
- px, em, rem, %
- auto

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
width: 200px;
height: 80px;
background: lightcoral;
border: 2px solid black;
}

.m-1 { margin: 20px; }
.m-2 { margin: 10px 40px; }
.m-3 { margin: 10px 20px 30px; }
.m-4 { margin: 5px 10px 15px 20px; }
.m-auto { margin: 20px auto; }
</style>
</head>
<body>

<div class="box m-1">margin: 20px</div>
<div class="box m-2">margin: 10px 40px</div>
<div class="box m-3">margin: 10px 20px 30px</div>
<div class="box m-4">margin: 5px 10px 15px 20px</div>
<div class="box m-auto">margin: auto (centered)</div>

</body>
</html>
```



How values are applied:

- 1 value → all sides
- 2 values → top/bottom | left/right
- 3 values → top | left/right | bottom
- 4 values → top | right | bottom | left

## Box Sizing

By default, width and height apply **only to content**.

box-sizing: content-box; **default**

box-sizing: border-box;

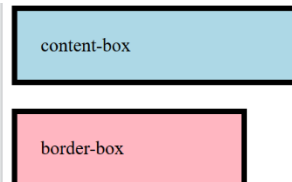
- content-box: padding and border add to size
- border-box: padding and border included in width/height (recommended)

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
  width: 200px;
  padding: 20px;
  border: 5px solid black;
  margin-bottom: 20px;
}

.content-box { box-sizing: content-box; background: lightblue; }
.border-box { box-sizing: border-box; background: lightpink; }
</style>
</head>
<body>

<div class="box content-box">content-box</div>
<div class="box border-box">border-box</div>

</body>
</html>
```



## Box Shadow

box-shadow adds **shadow effects** around elements to create depth, elevation, or emphasis. Shadows do **not** affect layout.

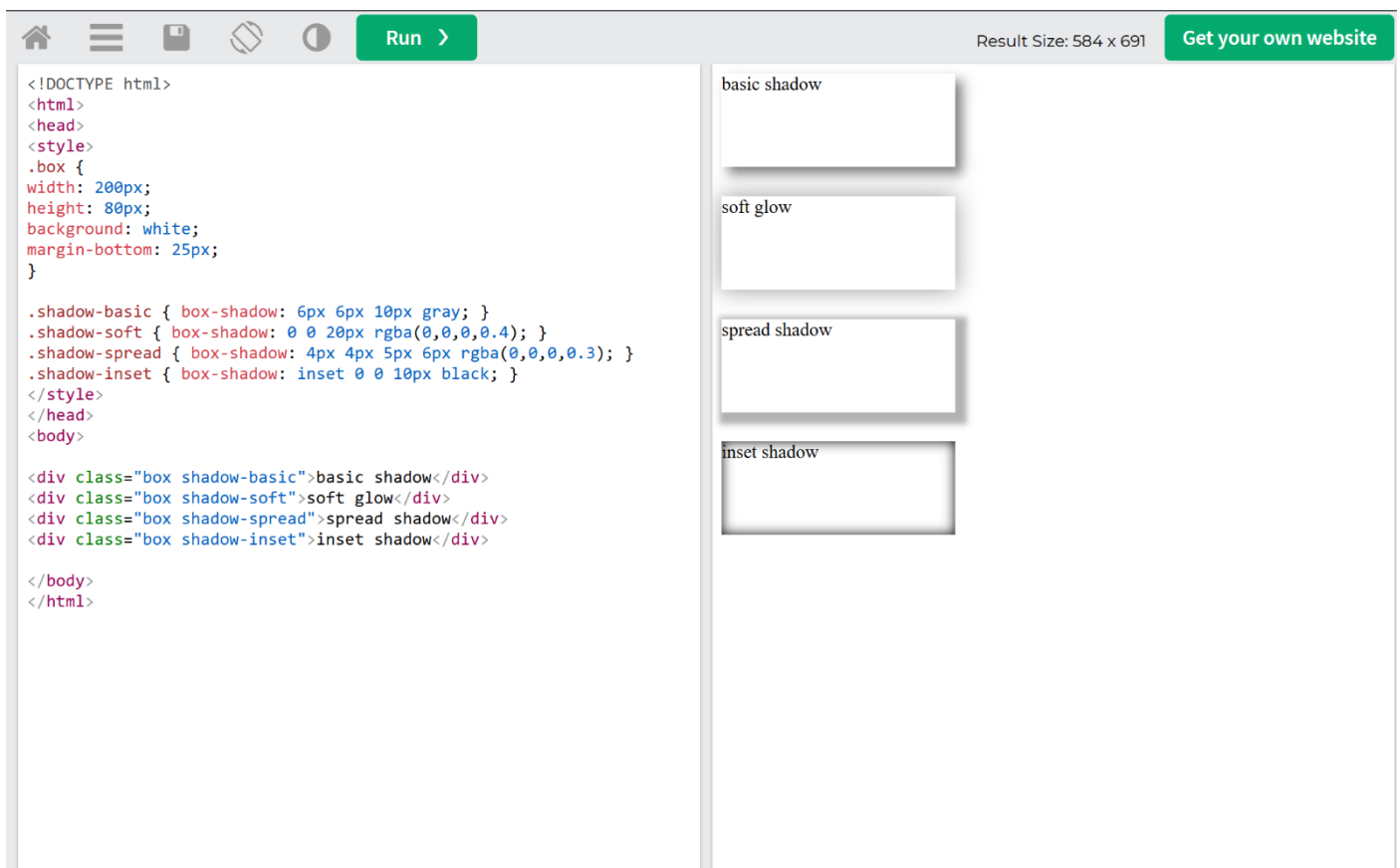
### Syntax:

box-shadow: offset-x offset-y blur-radius spread-radius color;

### Properties & values:

- offset-x → horizontal shift (required)
- offset-y → vertical shift (required)
- blur-radius → softness (optional)
- spread-radius → size expansion (optional)
- color → any CSS color format
- inset → shadow inside the element

Multiple shadows can be applied, separated by commas.



The screenshot shows a web browser interface with a code editor on the left and a preview area on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
width: 200px;
height: 80px;
background: white;
margin-bottom: 25px;
}

.shadow-basic { box-shadow: 6px 6px 10px gray; }
.shadow-soft { box-shadow: 0 0 20px rgba(0,0,0,0.4); }
.shadow-spread { box-shadow: 4px 4px 5px 6px rgba(0,0,0,0.3); }
.shadow-inset { box-shadow: inset 0 0 10px black; }
</style>
</head>
<body>

<div class="box shadow-basic">basic shadow</div>
<div class="box shadow-soft">soft glow</div>
<div class="box shadow-spread">spread shadow</div>
<div class="box shadow-inset">inset shadow</div>

</body>
</html>
```

The preview area displays four examples of box shadows:

- basic shadow: A white box with a gray drop shadow.
- soft glow: A white box with a soft, out-of-focus gray shadow.
- spread shadow: A white box with a gray shadow that expands to fill the space around it.
- inset shadow: A white box with a black shadow that is inset into the box.

# CSS Layout & Flow

## DISPLAY

---

### *What display controls*

- How an element participates in layout
  - Whether it starts on a new line
  - Whether width/height apply
- 

### *Common values*

- block
- inline
- inline-block
- none
- flex
- grid

Result Size: 584 x 691 [Get your own website](#)

```

<!doctype html>
<html>
<head>
<style>
.box {
background: lightblue;
padding: 10px;
margin: 5px;
width: 120px;
height: 60px;
border: 2px solid navy;
}

.block { display: block; }
.inline { display: inline; }
.inline-block { display: inline-block; }
.none { display: none; }
.flex { display: flex; gap: 10px; }
.grid { display: grid; grid-template-columns: repeat(3, 1fr); gap: 10px; }
</style>
</head>
<body>

<h3>Block</h3>
<div class="container">
<div class="box block">Block 1</div>
<div class="box block">Block 2</div>
</div>

<h3>Inline</h3>
<div class="container">
<div class="box inline">Inline 1</div>
<div class="box inline">Inline 2</div>
</div>

<h3>Inline-Block</h3>
<div class="container">
<div class="box inline-block">IB 1</div>
<div class="box inline-block">IB 2</div>
</div>

<h3>Display None</h3>
<div class="container">
<div class="box">Visible</div>
<div class="box none">Still here</div>
</div>

```

Result Size: 584 x 691 [Get your own website](#)

```

<div class="box inline-block">IB 1</div>
<div class="box inline-block">IB 2</div>
</div>

<h3>Inline-Block</h3>
<div class="container">
<div class="box inline-block">IB 1</div>
<div class="box inline-block">IB 2</div>
</div>

<h3>Display None</h3>
<div class="container">
<div class="box">Visible</div>
<div class="box none">Hidden</div>
<div class="box">Still here</div>
</div>

<h3>Flex (preview)</h3>
<div class="container flex">
<div class="box">1</div>
<div class="box">2</div>
<div class="box">3</div>
</div>

<h3>Grid (preview)</h3>
<div class="container grid">
<div class="box">A</div>
<div class="box">B</div>
<div class="box">C</div>
</div>

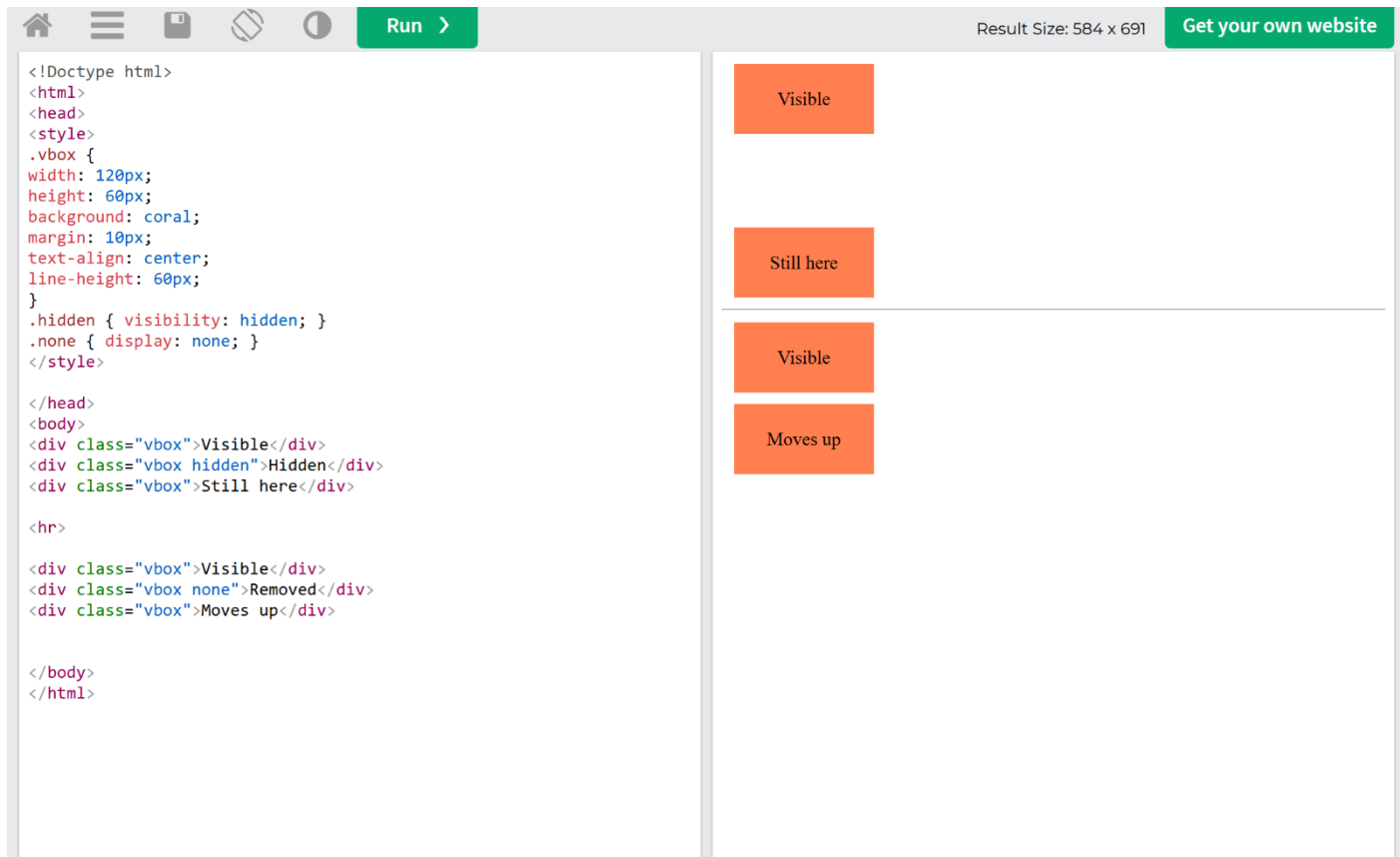
</body>
</html>

```

# VISIBILITY

## Key difference

- **visibility: hidden** → invisible **but space remains**
- **display: none** → removed completely



The screenshot shows a web browser interface with a code editor on the left and a rendered page on the right. The code editor contains the following HTML and CSS:

```
<!doctype html>
<html>
<head>
<style>
.vbox {
width: 120px;
height: 60px;
background: coral;
margin: 10px;
text-align: center;
line-height: 60px;
}
.hidden { visibility: hidden; }
.none { display: none; }
</style>

</head>
<body>
<div class="vbox">Visible</div>
<div class="vbox hidden">Hidden</div>
<div class="vbox">Still here</div>

<hr>

<div class="vbox">Visible</div>
<div class="vbox none">Removed</div>
<div class="vbox">Moves up</div>

</body>
</html>
```

The rendered page shows the following visual elements:

- Top section: An orange box labeled "Visible" is positioned above another orange box labeled "Still here".
- Bottom section (separated by a horizontal line): An orange box labeled "Visible" is positioned above another orange box labeled "Moves up".

The "Hidden" and "Removed" elements from the code are not visible in the rendered page, demonstrating the difference between the two CSS properties.

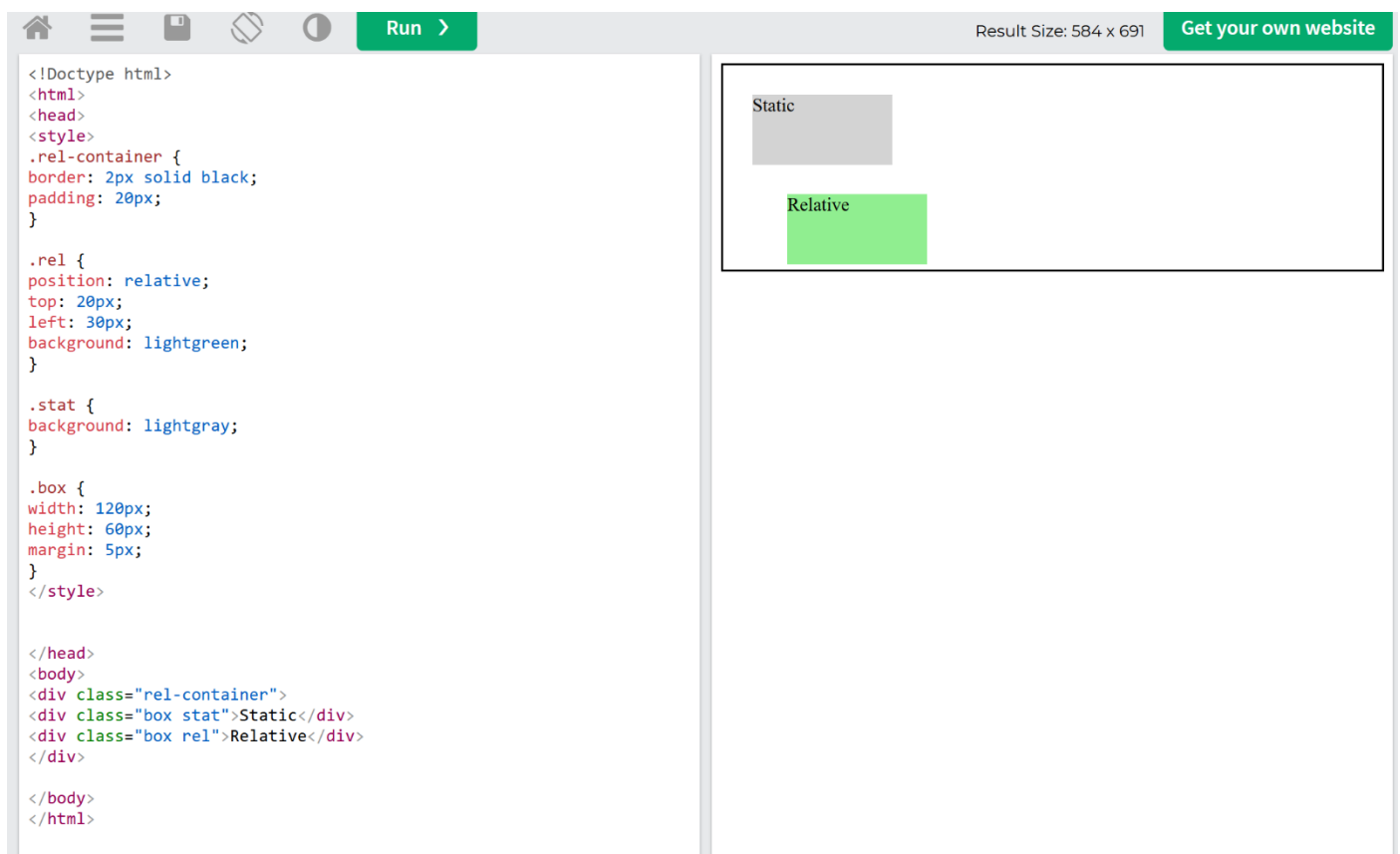
# POSITIONING

---

## Position types

- **static** (default)
  - **relative** (relative to Parent)
  - **absolute** (relative to Body)
  - **fixed** (relative to Viewport)
  - **sticky** (relative to the viewport until parent ends)
- 

## Static vs Relative



## Absolute with Positioned Parent

The screenshot shows a web browser interface with a code editor on the left and a preview window on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.parent {
position: relative;
width: 300px;
height: 200px;
border: 3px solid black;
}

.child {
position: absolute;
top: 20px;
right: 20px;
width: 100px;
height: 60px;
background: tomato;
}
</style>

</head>
<body>
<div class="parent">
Parent
<div class="child">Absolute</div>
</div>

</body>
</html>
```

The preview window shows a white page with a black-bordered box labeled "Parent" in the top-left corner. Inside the "Parent" box, there is a smaller red box labeled "Absolute" positioned in the top-right corner of the parent box.

## Absolute without Positioned Parent

The screenshot shows a web browser interface with a code editor on the left and a preview window on the right. The code editor contains the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.parent {
width: 300px;
height: 200px;
border: 3px solid black;
}

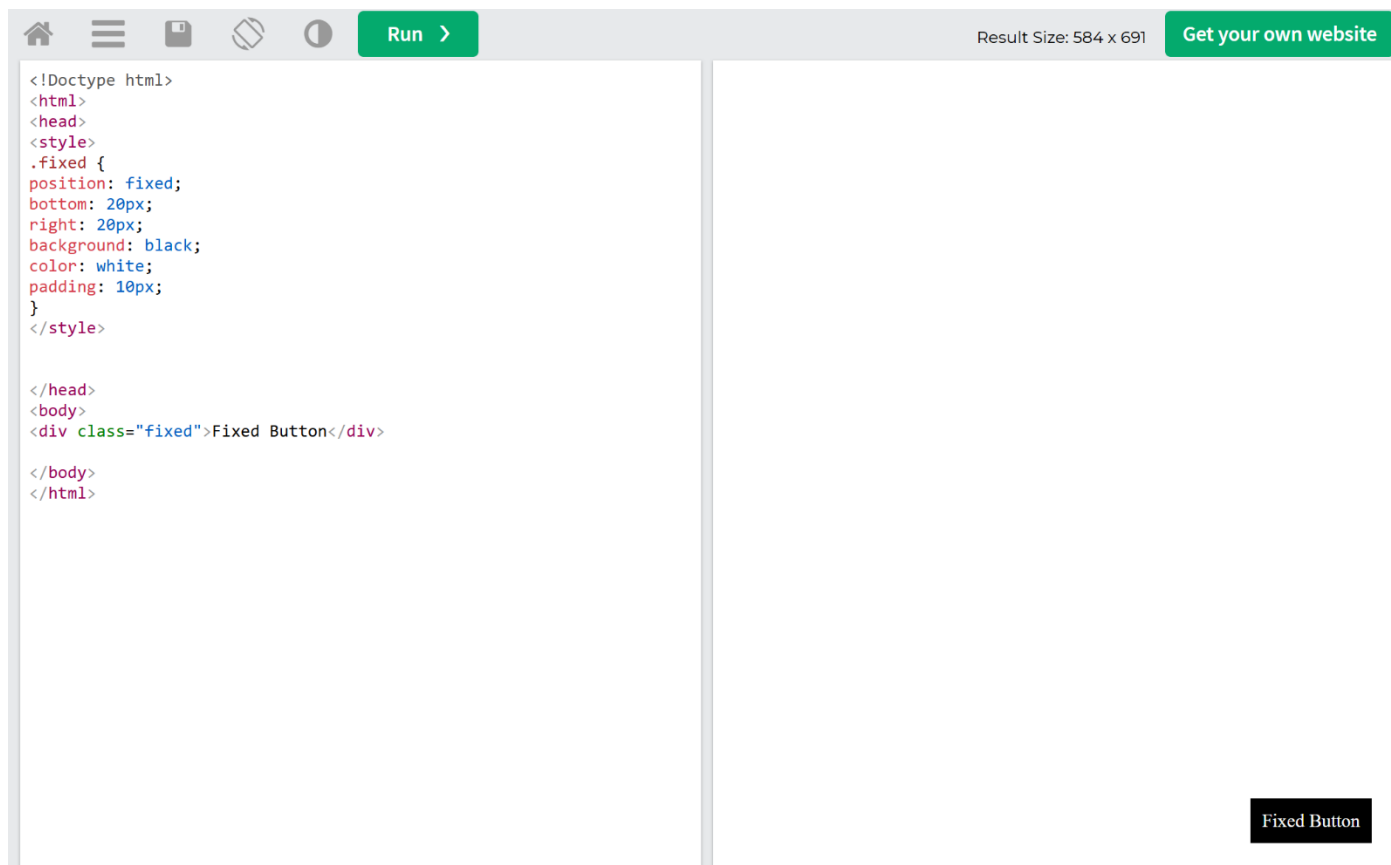
.child {
position: absolute;
top: 20px;
right: 20px;
width: 100px;
height: 60px;
background: tomato;
}
</style>

</head>
<body>
<div class="parent">
Parent
<div class="child">Absolute</div>
</div>

</body>
</html>
```

The preview window shows a white page with a black-bordered box labeled "Parent" in the top-left corner. The red box labeled "Absolute" is positioned in the top-right corner of the page, but it is not contained within the "Parent" box, demonstrating that it is not relative to the parent's position.

## Fixed Position



The screenshot shows a web browser interface with a code editor on the left and a rendered page on the right. The code editor contains the following HTML and CSS:

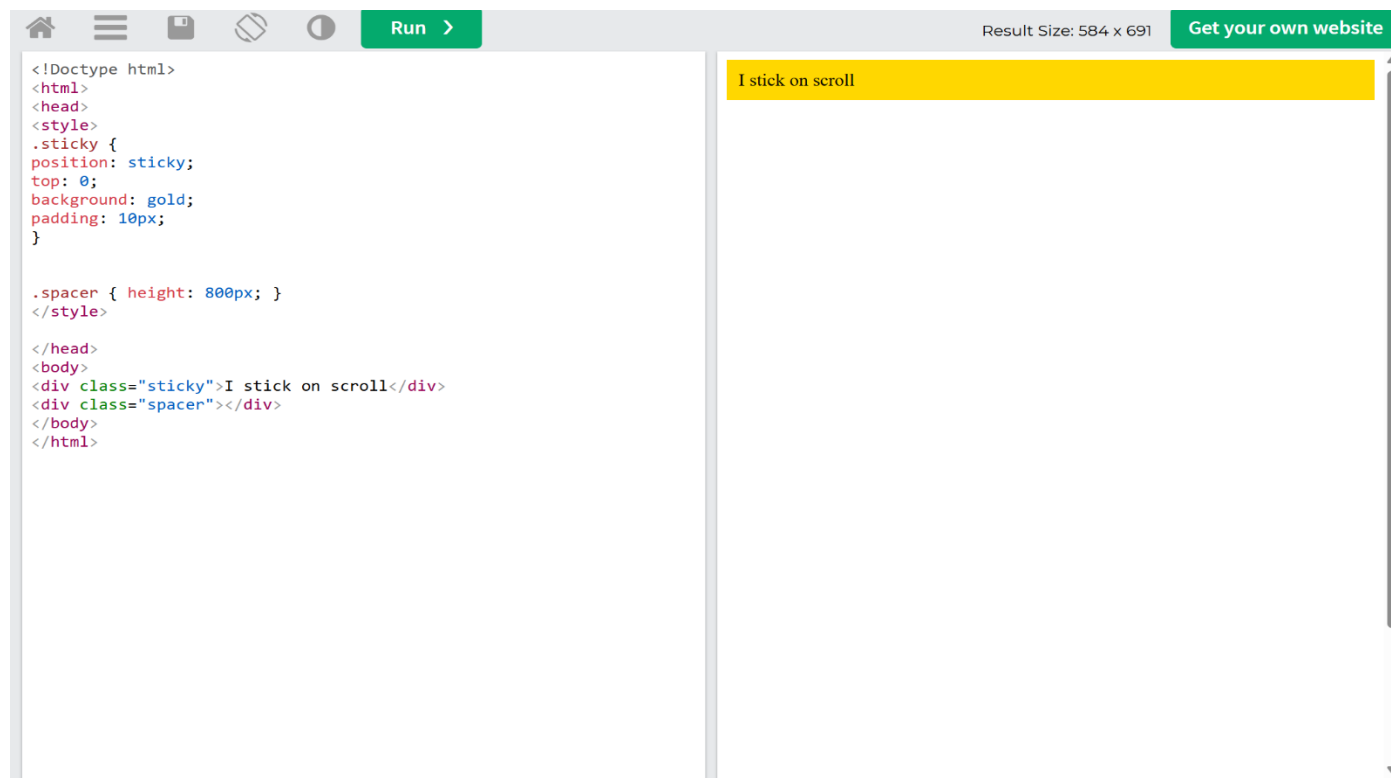
```
<!Doctype html>
<html>
<head>
<style>
.fixed {
position: fixed;
bottom: 20px;
right: 20px;
background: black;
color: white;
padding: 10px;
}
</style>

</head>
<body>
<div class="fixed">Fixed Button</div>

</body>
</html>
```

The rendered page shows a black button with the text "Fixed Button" in white, positioned at the bottom right of the page. The browser interface includes a "Run" button and a "Get your own website" link. The result size is 584 x 691.

## Sticky Position



The screenshot shows a web browser interface with a code editor on the left and a rendered page on the right. The code editor contains the following HTML and CSS:

```
<!Doctype html>
<html>
<head>
<style>
.sticky {
position: sticky;
top: 0;
background: gold;
padding: 10px;
}

.spacer { height: 800px; }
</style>

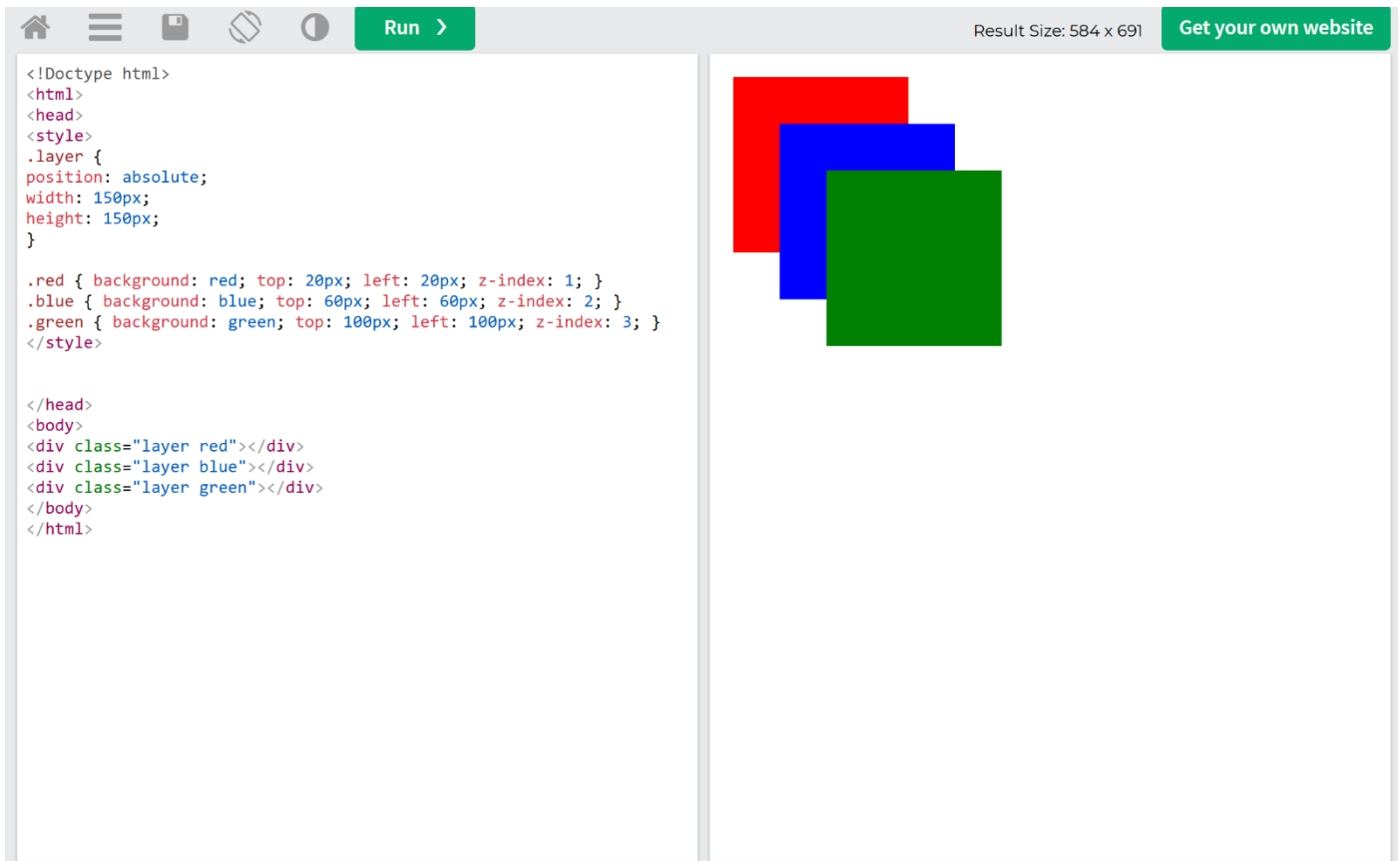
</head>
<body>
<div class="sticky">I stick on scroll</div>
<div class="spacer"></div>
</body>
</html>
```

The rendered page shows a yellow header with the text "I stick on scroll" that remains fixed at the top of the page as the user scrolls. Below the header is a large white area representing the spacer. The browser interface includes a "Run" button and a "Get your own website" link. The result size is 584 x 691.

# Z-INDEX & STACKING CONTEXT

## Rules

- Only works on positioned elements
- Higher z-index appears on top (0 is default), if indexes are equal -> first one created moves to back
- Parent stacking context limits children



The screenshot shows a web browser interface with a code editor on the left and a preview window on the right. The code editor contains the following HTML and CSS:

```
<!doctype html>
<html>
<head>
<style>
.layer {
position: absolute;
width: 150px;
height: 150px;
}

.red { background: red; top: 20px; left: 20px; z-index: 1; }
.blue { background: blue; top: 60px; left: 60px; z-index: 2; }
.green { background: green; top: 100px; left: 100px; z-index: 3; }
</style>

</head>
<body>
<div class="layer red"></div>
<div class="layer blue"></div>
<div class="layer green"></div>
</body>
</html>
```

The preview window displays three overlapping squares: a red square at the top-left, a blue square overlapping the red one, and a green square overlapping the blue one. The green square is the most prominent, demonstrating the effect of a higher z-index.

Result Size: 584 x 691

[Get your own website](#)

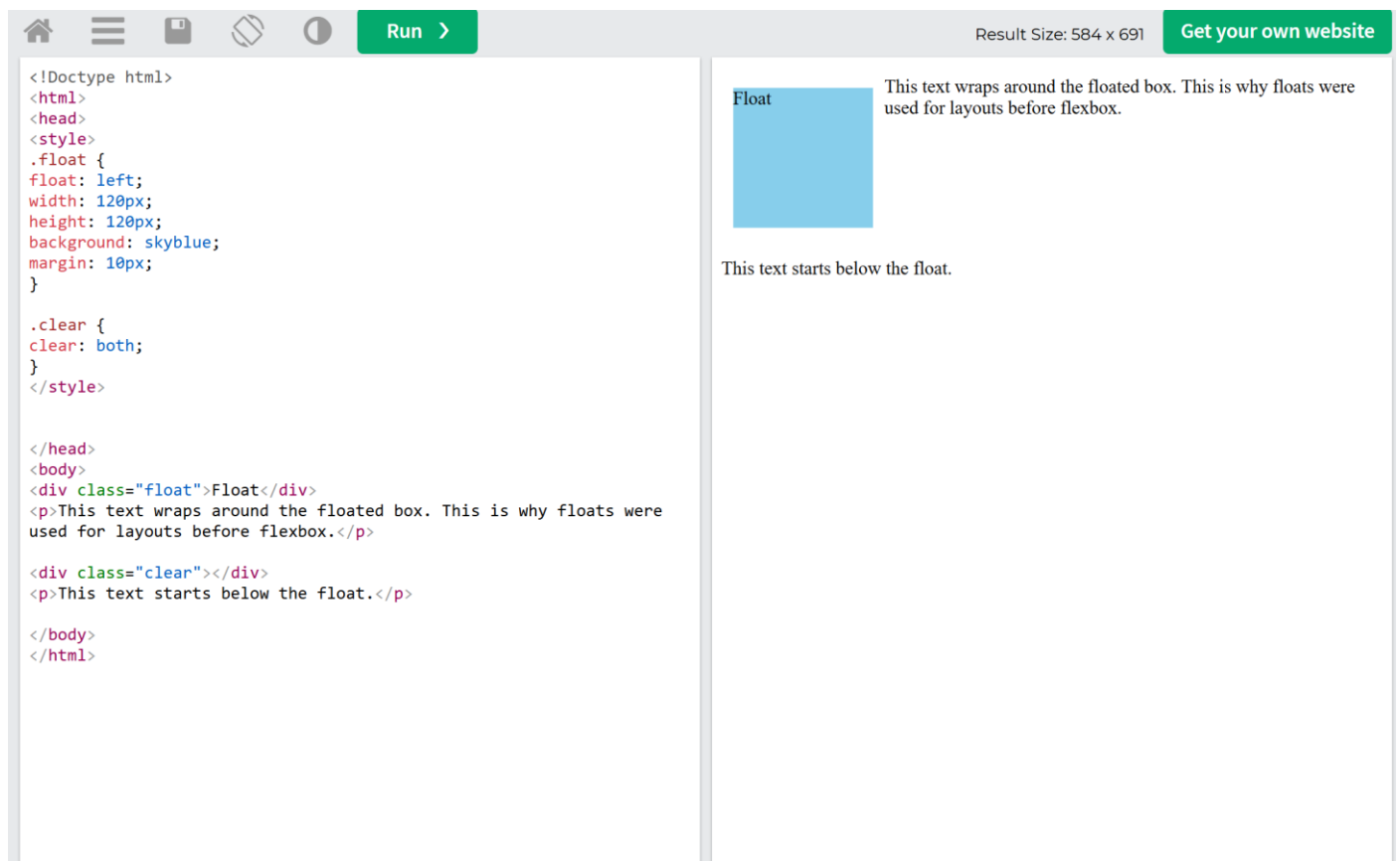
# FLOAT & CLEAR

## Float behavior

- Removes element from normal flow
- Text wraps around it
- Values of float: left, right, none(default)

## Clear behavior

- don't sit next to floated elements — start below them
- Values of clear: left, right, both, none(default)



The screenshot shows a web browser interface. On the left, there is a code editor with the following HTML code:

```
<!doctype html>
<html>
<head>
<style>
.float {
float: left;
width: 120px;
height: 120px;
background: skyblue;
margin: 10px;
}

.clear {
clear: both;
}
</style>

</head>
<body>
<div class="float">Float</div>
<p>This text wraps around the floated box. This is why floats were
used for layouts before flexbox.</p>

<div class="clear"></div>
<p>This text starts below the float.</p>

</body>
</html>
```

On the right, the rendered page shows a blue box labeled "Float" with the text "This text wraps around the floated box. This is why floats were used for layouts before flexbox." wrapping around it. Below the float, the text "This text starts below the float." is displayed.

# CSS Flexbox

Flexbox is a **one-dimensional layout system**. Each property below lists **all possible values**, then demonstrates **all of them together** using colored, numbered boxes so differences are obvious.

---

## Base Setup (Shared by All Examples)

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
body {
font-family: Arial, sans-serif;
}

.section {
margin-bottom: 60px;
}

.title {
font-weight: bold;
margin-bottom: 10px;
}

.container {
display: flex;
background: #eee;
padding: 10px;
margin-bottom: 15px;
border: 2px solid #ccc;
}

.box {
width: 70px;
height: 70px;
color: white;
font-size: 20px;
display: flex;
align-items: center;
justify-content: center;
}

.b1 { background: crimson; }
.b2 { background: dodgerblue; }
.b3 { background: seagreen; }
```

```
.b4 { background: goldenrod; }
</style>
</head>
<body>

</body>
</html>
```

## flex-direction

### Values

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

The screenshot shows a web browser interface with a code editor on the left and a preview area on the right. The code editor contains the following HTML and CSS:

```
justify-content: center;
}

.b1 { background: crimson; }
.b2 { background: dodgerblue; }
.b3 { background: seagreen; }
.b4 { background: goldenrod; }
</style>
</head>
<body>
<div class="section">
  <div class="title">row</div>
  <div class="container" style="flex-direction: row;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>

  <div class="title">row-reverse</div>
  <div class="container" style="flex-direction: row-reverse;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>

  <div class="title">column</div>
  <div class="container" style="flex-direction: column;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>

  <div class="title">column-reverse</div>
  <div class="container" style="flex-direction: column-reverse;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>
</div>
</body>
</html>
```

The preview area displays four examples of flex-direction values:

- row**: Three colored boxes (1, 2, 3) are arranged horizontally from left to right.
- row-reverse**: Three colored boxes (3, 2, 1) are arranged horizontally from right to left.
- column**: Three colored boxes (1, 2, 3) are arranged vertically from top to bottom.
- column-reverse**: Three colored boxes (3, 2, 1) are arranged vertically from bottom to top.

# flex-wrap

## Values

- nowrap (default)
- wrap
- wrap-reverse

The screenshot shows a web browser interface with a code editor on the left and a preview area on the right. The code editor contains the following HTML and CSS:

```
code { white-space: nowrap; font-size: 20px; display: flex; align-items: center; justify-content: center; }  
  
.b1 { background: crimson; }  
.b2 { background: dodgerblue; }  
.b3 { background: seagreen; }  
.b4 { background: goldenrod; }  
</style>  
</head>  
<body>  
<div class="section">  
<div class="title">nowrap</div>  
<div class="container" style="flex-wrap: nowrap; width: 200px;">  
<div class="box b1">1</div><div class="box b2">2</div><div class="box b3">3</div><div class="box b4">4</div>  
</div>  
  
<div class="title">wrap</div>  
<div class="container" style="flex-wrap: wrap; width: 200px;">  
<div class="box b1">1</div><div class="box b2">2</div><div class="box b3">3</div><div class="box b4">4</div>  
</div>  
  
<div class="title">wrap-reverse</div>  
<div class="container" style="flex-wrap: wrap-reverse; width: 200px;">  
<div class="box b1">1</div><div class="box b2">2</div><div class="box b3">3</div><div class="box b4">4</div>  
</div>  
</body>  
</html>
```

The preview area displays three examples of flex-wrap:

- nowrap:** A horizontal row of four colored boxes (1, 2, 3, 4) that do not wrap.
- wrap:** A 2x2 grid of colored boxes (1, 2, 3, 4) where the second row wraps.
- wrap-reverse:** A 2x2 grid of colored boxes (3, 4, 1, 2) where the second row wraps in reverse order.

# justify-content

## Values

- flex-start
- flex-end
- center
- space-between
- space-around
- space-evenly

The screenshot shows a web browser interface with a code editor on the left and a preview area on the right. The code editor contains HTML code for a section titled "justify-content" with six sub-sections, each demonstrating a different justify-content value. The preview area shows the corresponding visual results for each value: flex-start (items at the start), flex-end (items at the end), center (items centered), space-between (items with equal space between them), space-around (items with space around them), and space-evenly (items with even space around them).

```
<body>
<div class="section">
  <div class="title">flex-start</div>
  <div class="container" style="justify-content: flex-start;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>

  <div class="title">flex-end</div>
  <div class="container" style="justify-content: flex-end;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>

  <div class="title">center</div>
  <div class="container" style="justify-content: center;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>

  <div class="title">space-between</div>
  <div class="container" style="justify-content: space-between;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>

  <div class="title">space-around</div>
  <div class="container" style="justify-content: space-around;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>

  <div class="title">space-evenly</div>
  <div class="container" style="justify-content: space-evenly;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  </div>
</div>
</body>
```

flex-start

flex-end

center

space-between

space-around

space-evenly

# align-content (requires wrap)

## Values

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

```
<body>
<div class="section">
  <div class="title">align-content examples</div>

  <div class="example-title">align-content: stretch (default)</div>
  <div class="container" style="align-content: stretch;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
    <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
  </div>

  <div class="example-title">align-content: flex-start</div>
  <div class="container" style="align-content: flex-start;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
    <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
  </div>

  <div class="example-title">align-content: center</div>
  <div class="container" style="align-content: center;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
    <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
  </div>

  <div class="example-title">align-content: flex-end</div>
  <div class="container" style="align-content: flex-end;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
    <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
  </div>

  <div class="example-title">align-content: center</div>
  <div class="container" style="align-content: center;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
    <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
  </div>

  <div class="example-title">align-content: flex-end</div>
  <div class="container" style="align-content: flex-end;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
    <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
  </div>

  <div class="example-title">align-content: space-between</div>
  <div class="container" style="align-content: space-between;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
    <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
  </div>

  <div class="example-title">align-content: space-around</div>
  <div class="container" style="align-content: space-around;">
    <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
    <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
  </div>

  <div class="example-title">align-content: space-evenly</div>
```

```

<div class="example-title">align-content: flex-end</div>
<div class="container" style="align-content: flex-end;">
  <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
</div>

<div class="example-title">align-content: space-between</div>
<div class="container" style="align-content: space-between;">
  <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
</div>

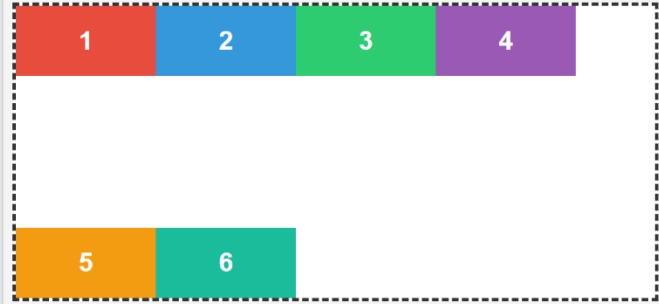
<div class="example-title">align-content: space-around</div>
<div class="container" style="align-content: space-around;">
  <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
</div>

<div class="example-title">align-content: space-evenly</div>
<div class="container" style="align-content: space-evenly;">
  <div class="box b1">1</div><div class="box b2">2</div><div
class="box b3">3</div>
  <div class="box b4">4</div><div class="box b5">5</div><div
class="box b6">6</div>
</div>

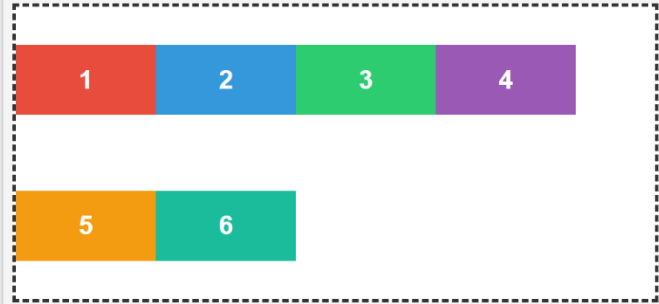
</div>
</body>
</html>

```

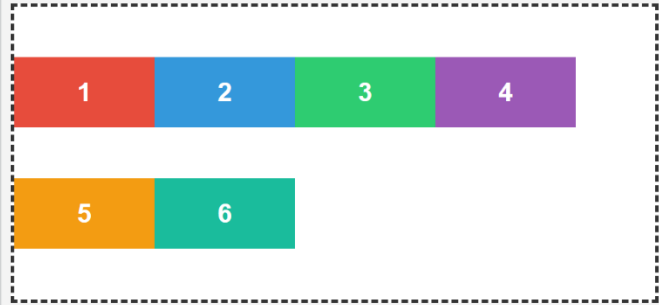
align-content: space-between



align-content: space-around



align-content: space-evenly



# align-items

## Values

- stretch (default)
- flex-start
- flex-end
- center
- baseline

Run >Result Size: 584 x 691Get your own website

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>align-items examples</title>

<style>
  body {
    font-family: Arial, sans-serif;
    background: #f4f4f4;
  }

  .section {
    margin-bottom: 60px;
  }

  .title {
    font-size: 22px;
    font-weight: bold;
    margin-bottom: 15px;
  }

  .example-title {
    margin: 15px 0 5px;
    font-weight: bold;
  }

  .container {
    display: flex;
    height: 150px;
    border: 3px dashed #333;
    margin-bottom: 25px;
    background: white;
  }

  .box {
    width: 120px;
    font-size: 22px;
    color: white;
  }

  .b1 { background: #e74c3c; }
  .b2 { background: #3498db; }
  .b3 { background: #2ecc71; }
</style>
</head>

<body>

<div class="section">
  <div class="title">align-items examples</div>

  <div class="example-title">align-items: stretch (default)</div>
  <div class="container stretch" style="align-items: stretch;">
    <div class="box b1">1</div>
    <div class="box b2">2</div>
    <div class="box b3">3</div>
  </div>

  <div class="example-title">align-items: flex-start</div>
  <div class="container" style="align-items: flex-start;">
    <div class="box b1" style="height: 60px;">1</div>
    <div class="box b2" style="height: 100px;">2</div>
    <div class="box b3" style="height: 80px;">3</div>
  </div>

  <div class="example-title">align-items: center</div>
  <div class="container" style="align-items: center;">
    <div class="box b1">1</div>
    <div class="box b2">2</div>
    <div class="box b3">3</div>
  </div>

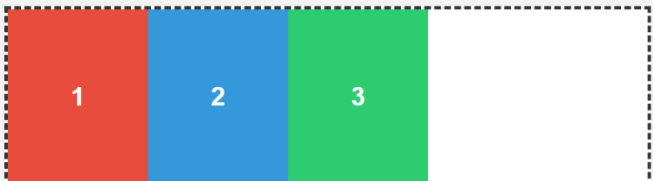
  <div class="example-title">align-items: flex-end</div>
  <div class="container" style="align-items: flex-end;">
    <div class="box b1">1</div>
    <div class="box b2">2</div>
    <div class="box b3">3</div>
  </div>

  <div class="example-title">align-items: flex-end</div>
  <div class="container" style="align-items: flex-end;">
    <div class="box b1">1</div>
    <div class="box b2">2</div>
    <div class="box b3">3</div>
  </div>

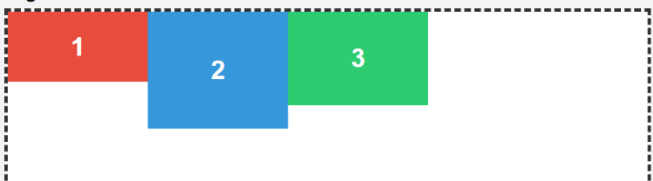
  <div class="example-title">align-items: baseline</div>
  <div class="container" style="align-items: baseline;">
    <div class="box b1">1</div>
    <div class="box b2">2</div>
    <div class="box b3">3</div>
  </div>
</div>
</body>
</html>
```

### align-items examples

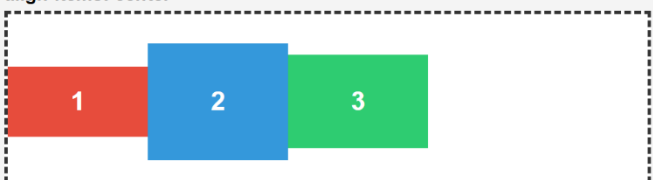
align-items: stretch (default)



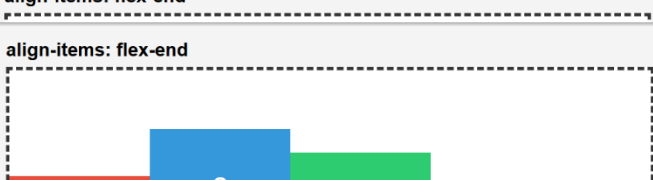
align-items: flex-start



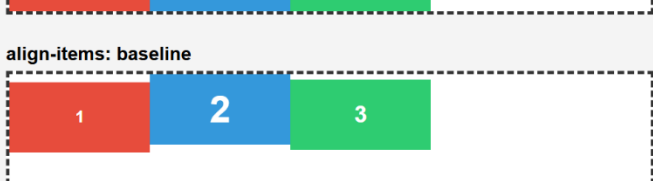
align-items: center



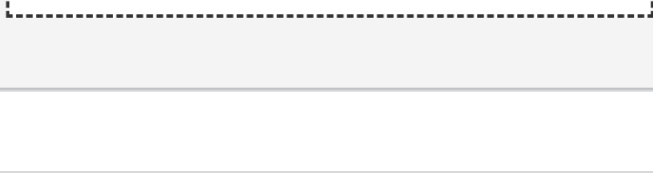
align-items: flex-end



align-items: flex-end



align-items: baseline



## align-self & justify-self

Align-self and justify-self use the same values as their container counterparts (align-items and justify-items), but they apply to a single element only.

They override the container's alignment for that specific item without affecting the rest.

```
.container {
  align-items: center;
}

.box.special {
  align-self: flex-end;
}
```

## Flex Item Properties (grow, shrink, basis, order)

### flex-grow

#### What it does

flex-grow controls how much an item grows relative to other items when there is extra space in the container.

- Default value: 0
- Works on the main axis
- Growth is proportional

#### Values

- 0 (do not grow)
- Any positive number (1, 2, 3, ...)

The screenshot shows a web browser window with a flex-grow example. The browser's address bar shows "Result Size: 591 x 691" and a "Get your own website" button. The main content area displays a flex container with three items: a red box labeled "grow: 1", a blue box labeled "grow: 2", and a green box labeled "grow: 0". The blue box is the largest, followed by the red box, and the green box is the smallest. The browser's developer tools are open, showing the CSS and HTML code for the example.

```
    }
    .box {
      height: 70px;
      width: 80px;
      color: white;
      font-weight: bold;
      display: flex;
      align-items: center;
      justify-content: center;
    }
    .b1 { background: crimson; }
    .b2 { background: dodgerblue; }
    .b3 { background: seagreen; }
  </style>
</head>
<body>
<div class="section">
<div class="title">flex-grow example (container has extra space)
</div>
<div class="container">
  <div class="box b1" style="flex-grow: 1;">grow: 1</div>
  <div class="box b2" style="flex-grow: 2;">grow: 2</div>
  <div class="box b3" style="flex-grow: 0;">grow: 0</div>
</div>
</body>
</html>
```

## flex-shrink

### What it does

flex-shrink controls **how much an item shrinks** when there is **not enough space** in the container.

- Default value: 1
- Works only when items overflow
- Shrinking is **proportional**

### Values

- 0 (do not shrink)
- Any positive number (1, 2, ...)

```
    }  
    }  
    .box {  
      height: 70px;  
      width: 80px;  
      color: white;  
      font-weight: bold;  
      display: flex;  
      align-items: center;  
      justify-content: center;  
    }  
    .b1 { background: crimson; }  
    .b2 { background: dodgerblue; }  
    .b3 { background: seagreen; }  
  </style>  
</head>  
<body>  
<div class="section">  
  <div class="title">flex-shrink example (container is too small)  
</div>  
  <div class="container" style="width: 300px;">  
    <div class="box b1" style="width: 200px; flex-shrink:  
1;">shrink: 1</div>  
    <div class="box b2" style="width: 200px; flex-shrink:  
0;">shrink: 0</div>  
    <div class="box b3" style="width: 200px; flex-shrink:  
1;">shrink: 1</div>  
  </div>  
</body>  
</html>
```

flex-shrink example (container is too small)

shrink: 1      shrink: 0      shrink: 1

## flex-basis

### What it does

flex-basis defines the **initial size** of a flex item **before grow or shrink happens**.

Think of it as:

“My starting width (or height)”

### Values

- Lengths: px, %, em, etc.
- auto (default)
- content

The screenshot shows a web development tool interface. On the left is a code editor with the following CSS and HTML code:

```
background-color: #ccc;
}

.box {
  height: 70px;
  width: 80px;
  color: white;
  font-weight: bold;
  display: flex;
  align-items: center;
  justify-content: center;
}

.b1 { background: crimson; }
.b2 { background: dodgerblue; }
.b3 { background: seagreen; }
</style>

</head>

<body>
<div class="section">
  <div class="title">flex-basis example (initial size)</div>

  <div class="container">
    <div class="box b1" style="flex-basis: 100px;">basis:
100px</div>
    <div class="box b2" style="flex-basis: 200px;">basis:
200px</div>
    <div class="box b3" style="flex-basis: auto;">basis:
auto</div>
  </div>
</div>

</body>
</html>
```

On the right is a visual preview of the code. It shows a container with a dashed border containing three colored boxes: a red box labeled "basis: 100px", a blue box labeled "basis: 200px", and a green box labeled "basis: auto". The text "flex-basis example (initial size)" is centered above the boxes. The tool's top bar shows "Result Size: 591 x 691" and a "Get your own website" button.

## order

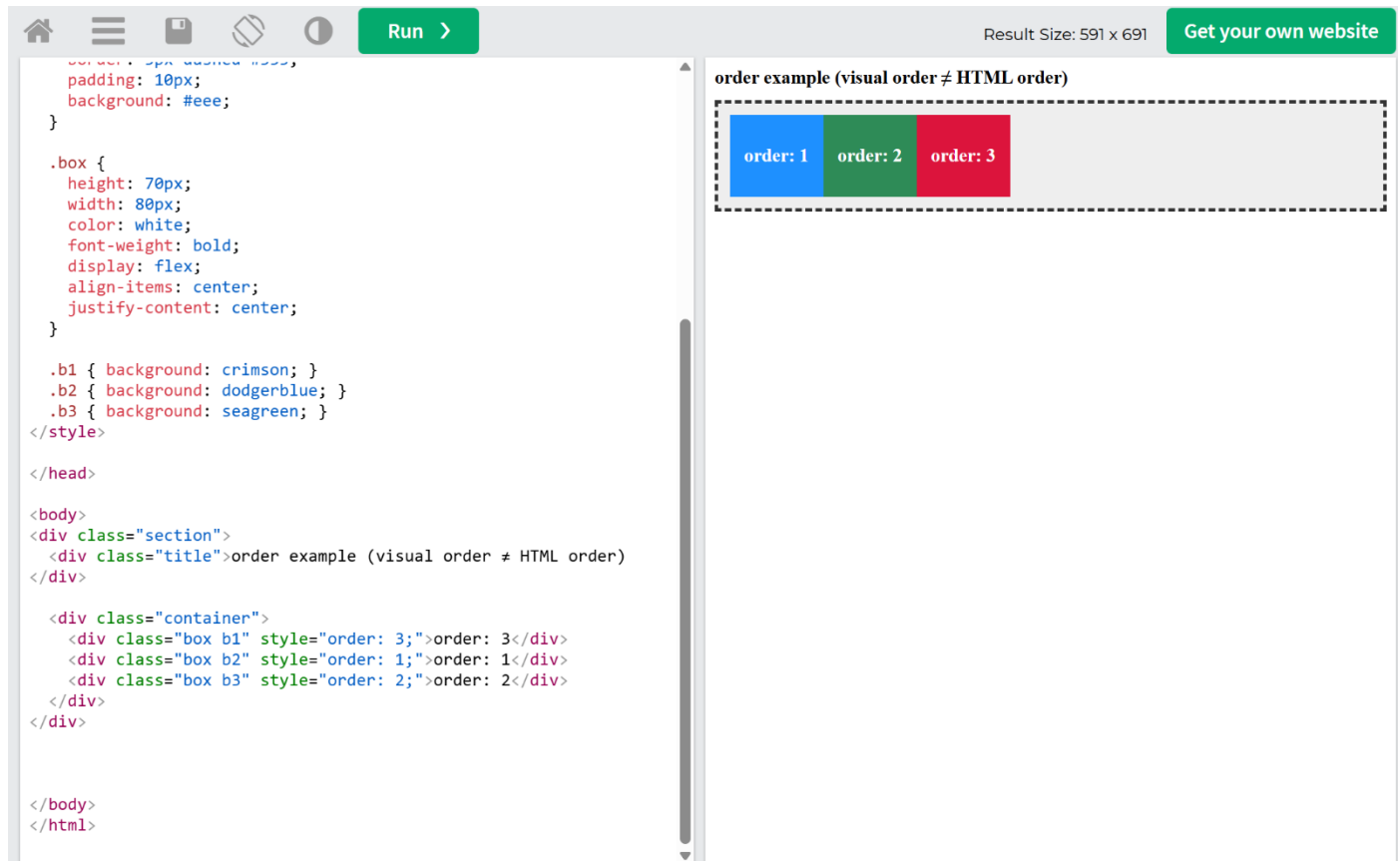
### What it does

order controls the **visual order** of items **without changing HTML order**.

- Default value: 0
- Lower numbers appear first

### Values

- Any integer (-1, 0, 1, 10, etc.)



The screenshot shows a web browser interface with a code editor on the left and a rendered preview on the right. The code editor contains the following HTML and CSS:

```
.border { border: 1px dashed #eee; padding: 10px; background: #eee; }

.box { height: 70px; width: 80px; color: white; font-weight: bold; display: flex; align-items: center; justify-content: center; }

.b1 { background: crimson; }
.b2 { background: dodgerblue; }
.b3 { background: seagreen; }
</style>

</head>

<body>
<div class="section">
  <div class="title">order example (visual order ≠ HTML order)
</div>

  <div class="container">
    <div class="box b1" style="order: 3;">order: 3</div>
    <div class="box b2" style="order: 1;">order: 1</div>
    <div class="box b3" style="order: 2;">order: 2</div>
  </div>
</div>

</body>
</html>
```

The rendered preview shows a dashed box containing three colored boxes: a blue box labeled "order: 1", a green box labeled "order: 2", and a red box labeled "order: 3". The boxes are arranged in the order specified by their 'order' attribute, demonstrating that the visual order is not necessarily the same as the HTML order.

# CSS Grid Layout

CSS Grid is a **2D layout system** that lets you control **rows and columns at the same time**.

## Grid Basics (must exist first)

```
.container {  
  display: grid;  
}
```

Nothing works without this.

## grid-template-columns

### What it does

Defines **how many columns** the grid has and their sizes.

### Common values

- Fixed: 100px, 200px
- Flexible: 1fr, 2fr -> **by ratio**
- Auto: auto
- Functions:
  - repeat(4, 25% 25%) -> **repeat four times, each time 25 %, resulting in 8 columns each 25%**
  - minmax(**minimum value, maximum value**)
  - fit-content(200px) -> fits data in 200px if it is more, it will wrap in the next line.

The screenshot shows a web browser window with a code editor on the left and a preview on the right. The code editor contains the following CSS and HTML:

```
/* GRID TEMPLATES */  
.columns {  
  grid-template-columns: 100px 1fr 2fr;  
}  
  
/* GRID ITEMS */  
.box {  
  height: 70px;  
  color: white;  
  font-weight: bold;  
  display: flex;  
  align-items: center;  
  justify-content: center;  
}  
  
.b1 { background: crimson; }  
.b2 { background: dodgerblue; }  
.b3 { background: seagreen; }  
.b4 { background: purple; }  
  
</style>  
</head>  
<body>  
<div class="section">  
<div class="title">grid-template-columns: 100px 1fr 2fr</div>  
  
<div class="grid columns">  
<div class="box b1">1</div>  
<div class="box b2">2</div>  
<div class="box b3">3</div>  
<div class="box b4">4</div>  
</div>  
</div>  
</body>
```

The preview shows the result of the CSS Grid layout. The title is "grid-template-columns: 100px 1fr 2fr". The grid contains four boxes: a red box with the number "1", a blue box with the number "2", a green box with the number "3", and a purple box with the number "4". The red, blue, and green boxes are in the first row, and the purple box is in the second row, aligned to the left.

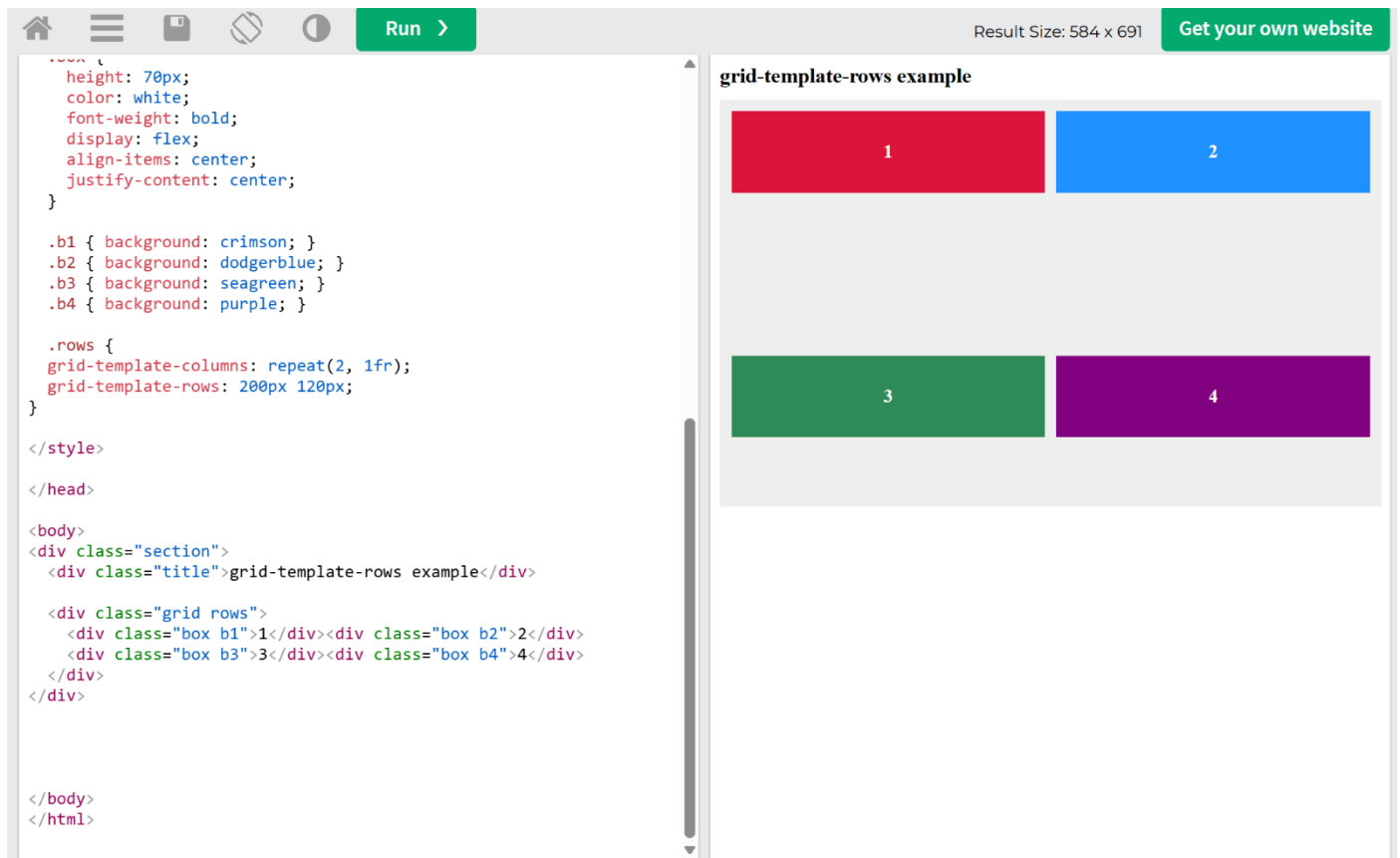
# grid-template-rows

## What it does

Defines **row heights**.

## Values

- Same as columns: px, fr, auto, minmax()



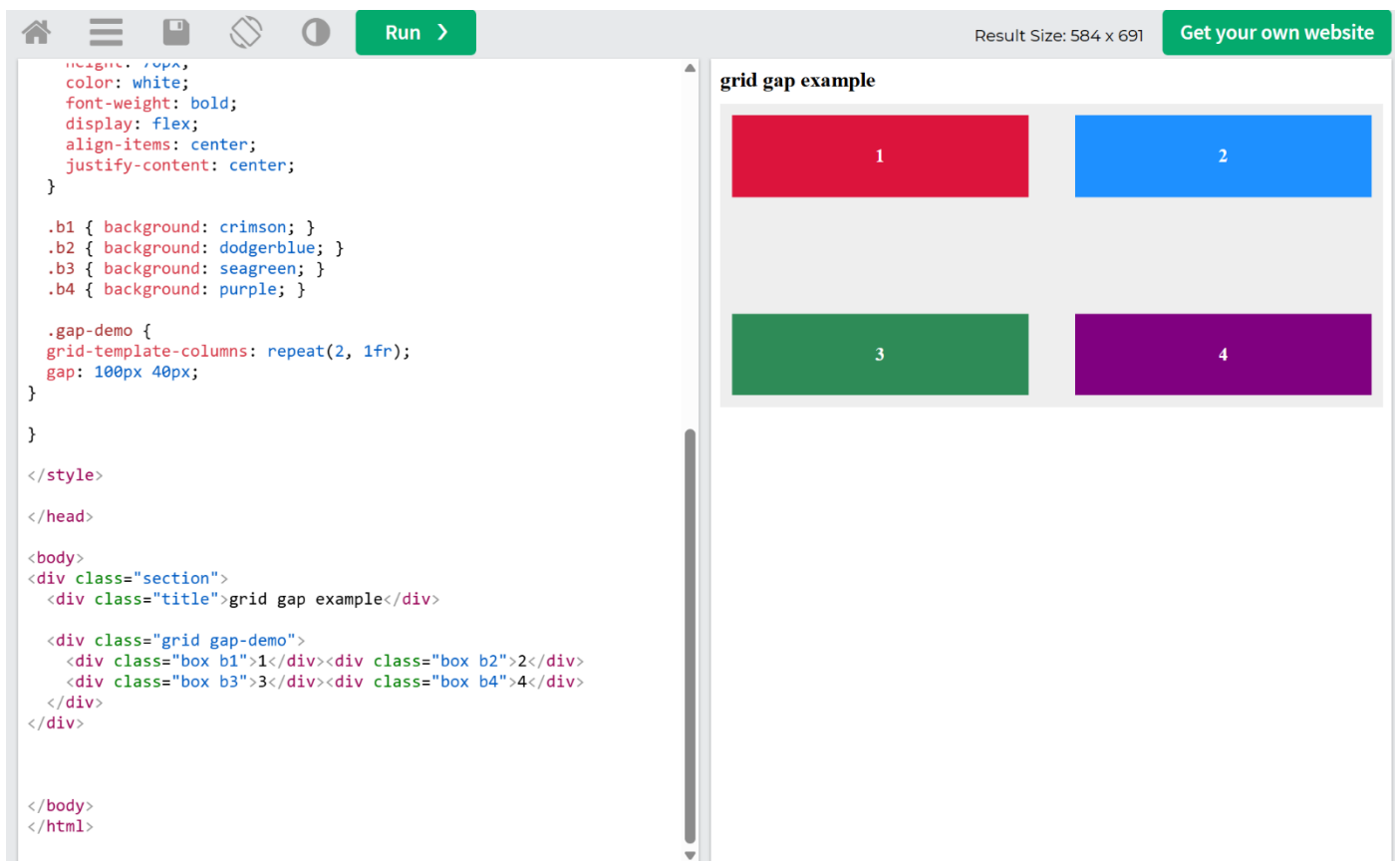
# gap (row-gap / column-gap)

## What it does

Controls spacing between grid cells.

## Values

- gap
- row-gap
- column-gap



## *Grid Alignment (Container)*

**justify-items** (horizontal alignment of items)

**align-items** (vertical alignment of items)

**Values**

- start
  - center
  - end
  - stretch (default)
- 

## *Grid Content Alignment (whole grid)*

**justify-content**

**align-content**

These move the **entire grid inside the container**.

**Values**

- start
  - center
  - end
  - space-between
  - space-around
  - space-evenly
- 

## *Grid Item Alignment (per item)*

**justify-self**

**align-self**

Same idea as Flexbox:

**Overrides container alignment for one item**

## Grid Lines (Positioning by lines)

### What are Grid Lines?

When you create a CSS Grid, the **columns and rows create invisible lines**:

- Vertical lines → column lines
- Horizontal lines → row lines

These lines are **numbered automatically** starting from **1**, counting from the **top-left** corner:

- Columns: left → right
- Rows: top → bottom

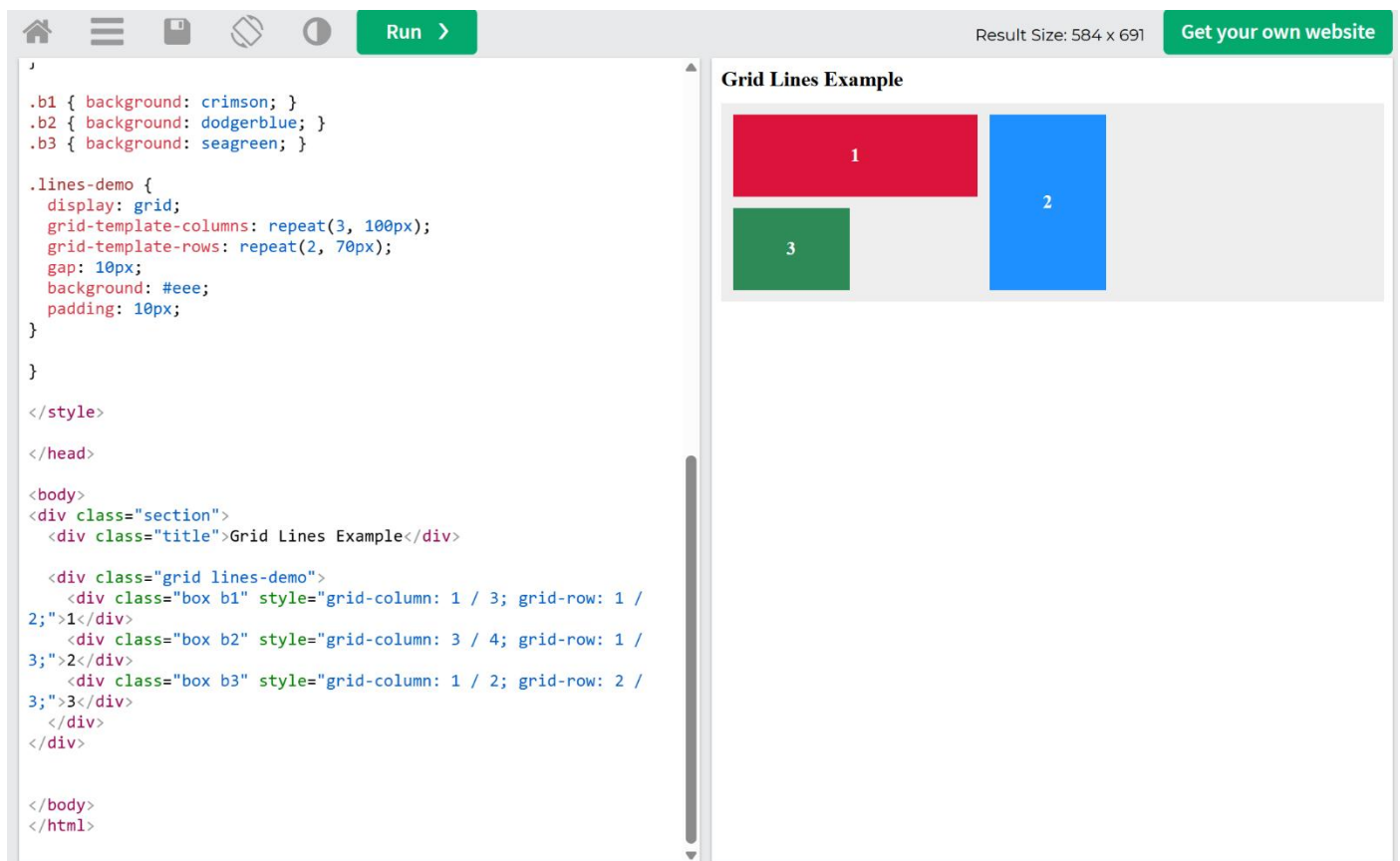
You can **position items by referring to these lines** using:

grid-column: start / end;

grid-row: start / end;

Where:

- start = line number to start at
- end = line number to end **before** that line



The screenshot shows a web browser interface with a code editor on the left and a preview on the right. The code editor contains the following CSS and HTML:

```
.b1 { background: crimson; }
.b2 { background: dodgerblue; }
.b3 { background: seagreen; }

.lines-demo {
  display: grid;
  grid-template-columns: repeat(3, 100px);
  grid-template-rows: repeat(2, 70px);
  gap: 10px;
  background: #eee;
  padding: 10px;
}

</style>

</head>

<body>
<div class="section">
  <div class="title">Grid Lines Example</div>

  <div class="grid lines-demo">
    <div class="box b1" style="grid-column: 1 / 3; grid-row: 1 / 2;">1</div>
    <div class="box b2" style="grid-column: 3 / 4; grid-row: 1 / 3;">2</div>
    <div class="box b3" style="grid-column: 1 / 2; grid-row: 2 / 3;">3</div>
  </div>
</div>

</body>
</html>
```

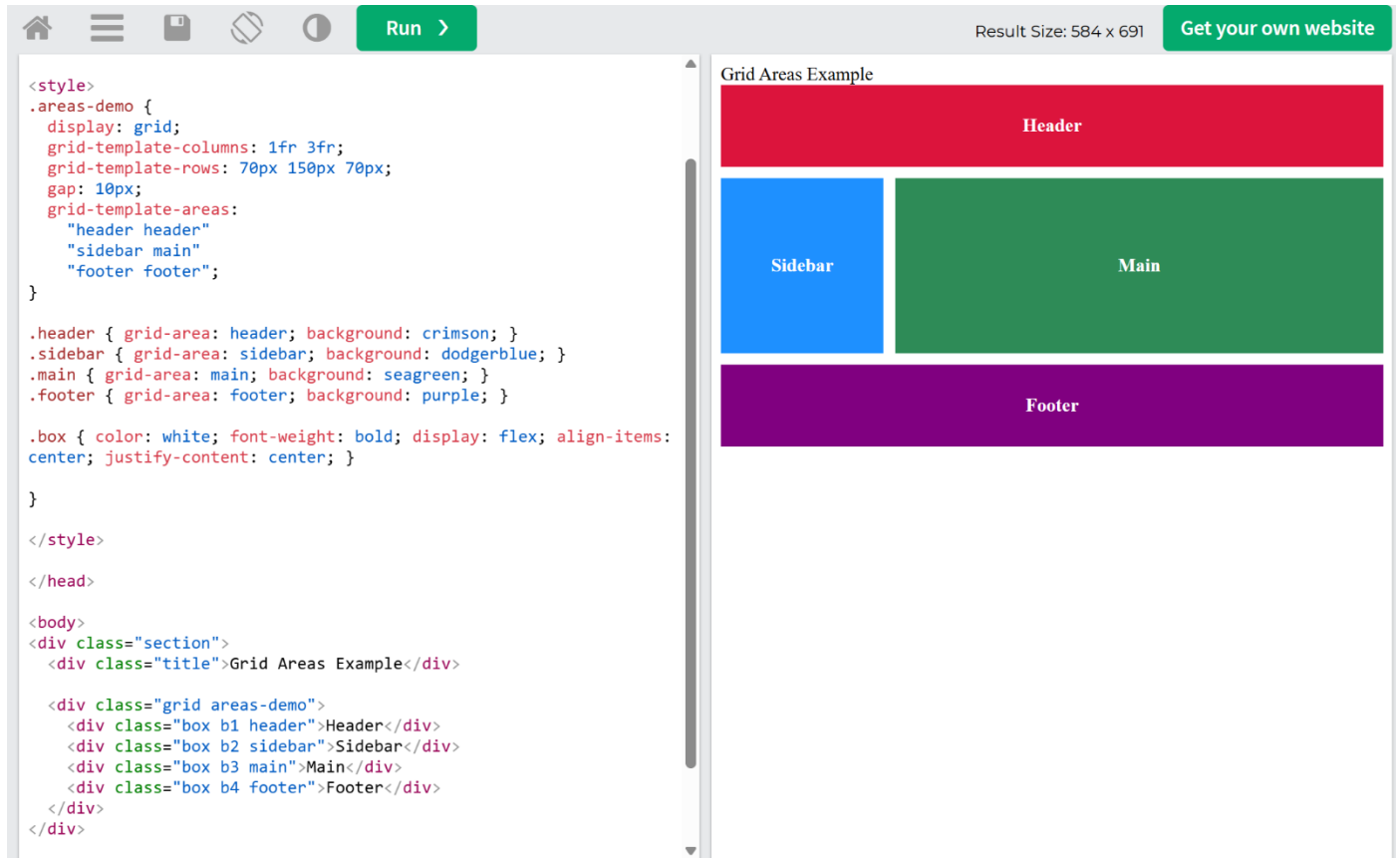
The preview on the right, titled "Grid Lines Example", shows a grid with three colored boxes: a red box labeled "1" in the top-left, a blue box labeled "2" in the top-right, and a green box labeled "3" in the bottom-left. The grid has a light gray background and a 10px gap between the boxes.

## Grid Areas (named layouts)

### What are Grid Areas?

Instead of using numbers, you can give parts of the grid a name and assign items to those areas.

- Makes your layout more readable and maintainable
- Great for complex page layouts like header, sidebar, main, footer



The screenshot shows a web browser window with a code editor on the left and a rendered page on the right. The code editor contains CSS and HTML code for a grid layout. The rendered page, titled "Grid Areas Example", shows a layout with four named areas: a red "Header" at the top, a blue "Sidebar" on the left, a green "Main" area on the right, and a purple "Footer" at the bottom. The browser interface includes a "Run" button, a "Result Size: 584 x 691" indicator, and a "Get your own website" button.

```
<style>
.areas-demo {
  display: grid;
  grid-template-columns: 1fr 3fr;
  grid-template-rows: 70px 150px 70px;
  gap: 10px;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
}

.header { grid-area: header; background: crimson; }
.sidebar { grid-area: sidebar; background: dodgerblue; }
.main { grid-area: main; background: seagreen; }
.footer { grid-area: footer; background: purple; }

.box { color: white; font-weight: bold; display: flex; align-items:
center; justify-content: center; }
</style>

</head>

<body>
<div class="section">
  <div class="title">Grid Areas Example</div>

  <div class="grid areas-demo">
    <div class="box b1 header">Header</div>
    <div class="box b2 sidebar">Sidebar</div>
    <div class="box b3 main">Main</div>
    <div class="box b4 footer">Footer</div>
  </div>
</div>
```

# CSS Backgrounds

The background property is actually a **shorthand** for several background properties.

## *background-color*

### What it does

Sets the **background color** of an element.

### Values

- Named colors: red, blue, yellow
- Hex: #ff0000
- RGB: rgb(255,0,0)
- RGBA: rgba(255,0,0,0.5)
- HSL: hsl(0, 100%, 50%)
- Transparent: transparent

The screenshot shows a web browser interface. On the left, there is a code editor with the following CSS and HTML code:

```
.title {
  font-weight: bold;
  margin-bottom: 10px;
  font-size: 18px;
}

.box {
  color: white;
  font-weight: bold;
  display: flex;
  align-items: center;
  justify-content: center;
}

</style>
</head>
<body>
<div class="section">
  <div class="title">background-color examples</div>

  <div class="box" style="background-color: red;">red</div>
  <div class="box" style="background-color: rgb(0, 128, 255);">rgb(0,128,255)</div>
  <div class="box" style="background-color: rgba(0, 128, 255, 0.5);">rgba(0,128,255,0.5)</div>
  <div class="box" style="background-color: hsl(120, 60%, 50%);">hsl(120,60%,50%)</div>
</div>
</body>
</html>
```

On the right, there is a preview area titled "background-color examples" showing four horizontal bars with the following text and colors:

Color
red
rgb(0,128,255)
rgba(0,128,255,0.5)
hsl(120,60%,50%)

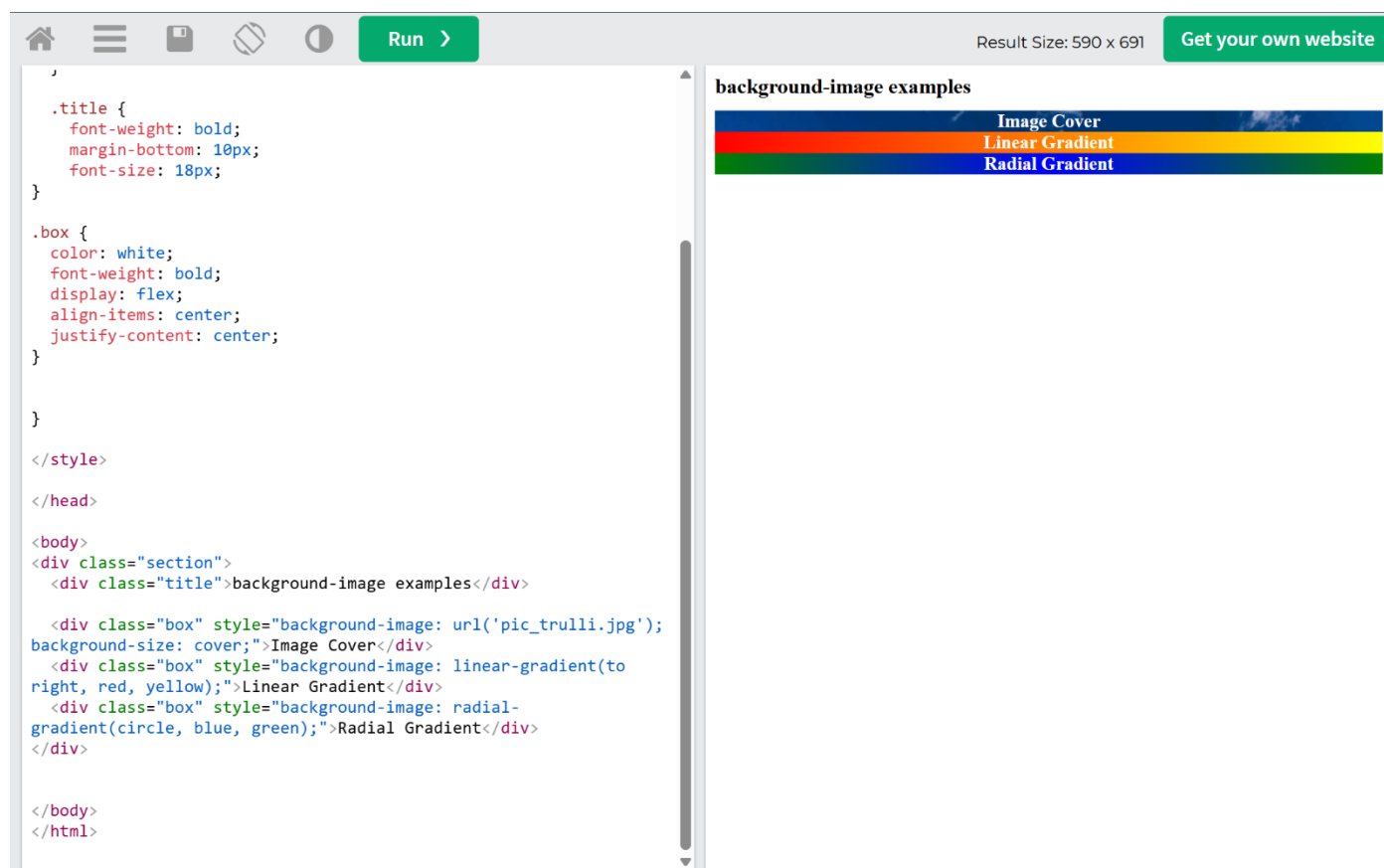
## background-image

### What it does

Sets an **image** as the background.

### Values

- url("image.jpg")
- linear-gradient(to right or angle ..., red, blue ...)
- radial-gradient(circle or ellipse ..., red, yellow, green ...)
- Multiple images separated by commas



The screenshot shows a web development tool interface. On the left is a code editor with the following HTML and CSS code:

```
.title {
  font-weight: bold;
  margin-bottom: 10px;
  font-size: 18px;
}

.box {
  color: white;
  font-weight: bold;
  display: flex;
  align-items: center;
  justify-content: center;
}

</style>
</head>
<body>
<div class="section">
  <div class="title">background-image examples</div>

  <div class="box" style="background-image: url('pic_trulli.jpg');
background-size: cover;">Image Cover</div>
  <div class="box" style="background-image: linear-gradient(to
right, red, yellow);">Linear Gradient</div>
  <div class="box" style="background-image: radial-
gradient(circle, blue, green);">Radial Gradient</div>
</div>

</body>
</html>
```

On the right is a preview window titled "background-image examples" showing three horizontal bars:

- Image Cover: A bar with a landscape image background.
- Linear Gradient: A bar with a gradient from red to yellow.
- Radial Gradient: A bar with a gradient from blue to green.

The tool interface includes a "Run" button and a "Get your own website" button. The result size is 590 x 691.

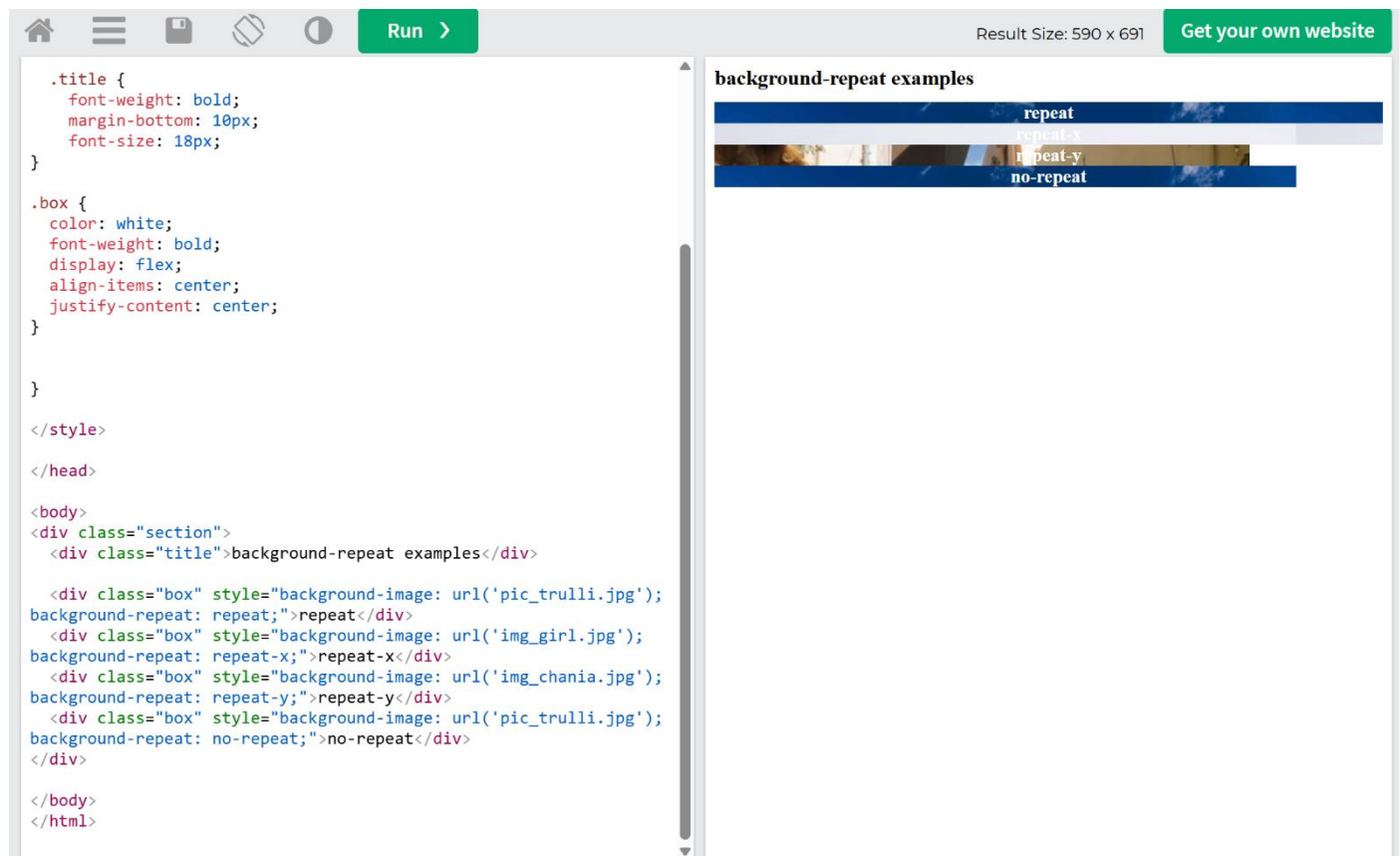
# background-repeat

## What it does

Controls if/how the background image repeats.

## Values

- repeat (default) → both X & Y
- repeat-x → horizontal only
- repeat-y → vertical only
- no-repeat → no repeat



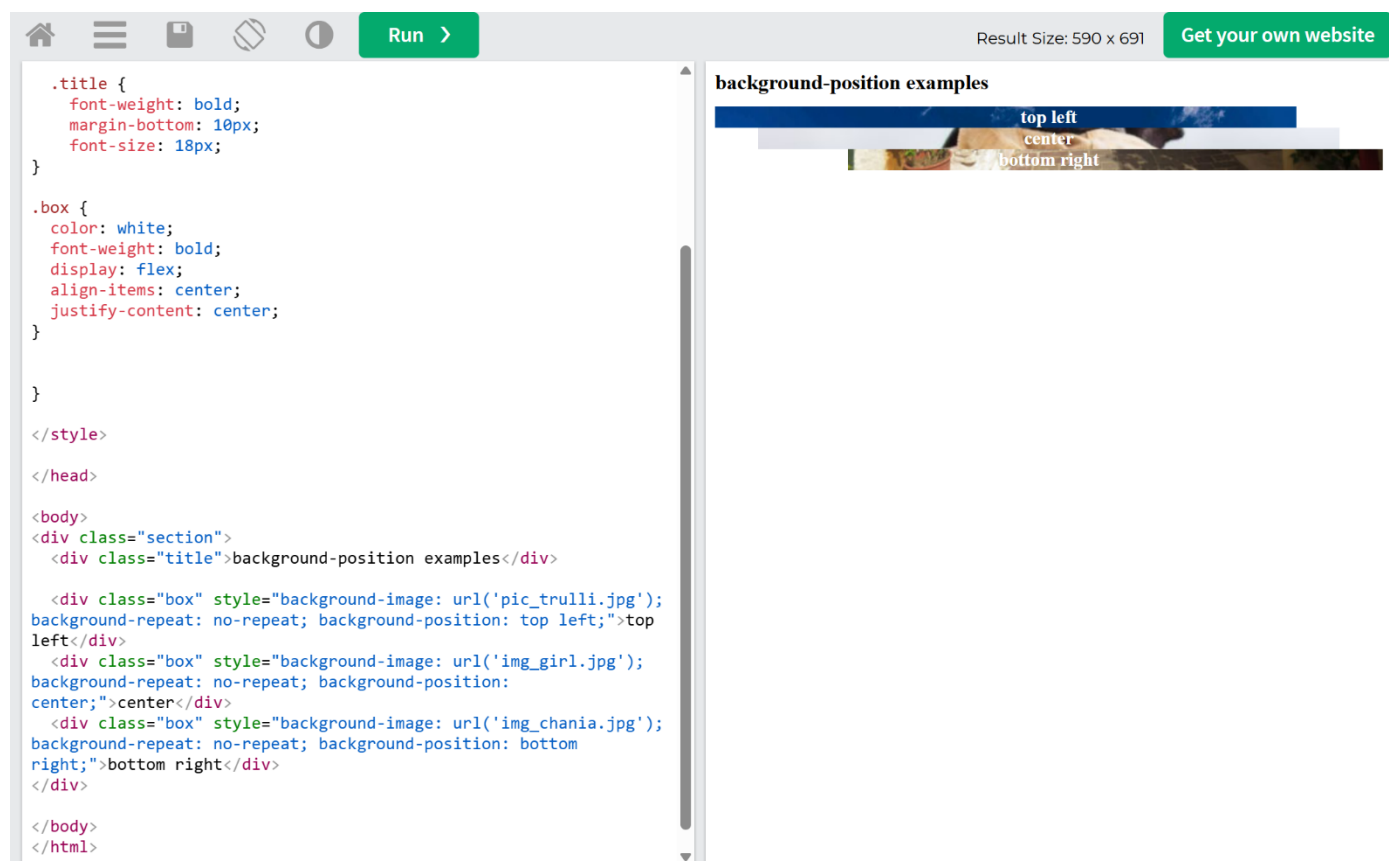
## background-position

### What it does

Sets **where the image starts** in the element.

### Values

- Keywords: top, bottom, left, right, center, top left ...
- Lengths: px, %
- Example: background-position: right 10px top 20px;



The screenshot shows a web browser interface with a code editor on the left and a preview on the right. The code editor contains the following HTML and CSS:

```
.title {
  font-weight: bold;
  margin-bottom: 10px;
  font-size: 18px;
}

.box {
  color: white;
  font-weight: bold;
  display: flex;
  align-items: center;
  justify-content: center;
}

</style>
</head>
<body>
<div class="section">
  <div class="title">background-position examples</div>
  <div class="box" style="background-image: url('pic_trulli.jpg');
background-repeat: no-repeat; background-position: top left;">top
left</div>
  <div class="box" style="background-image: url('img_girl.jpg');
background-repeat: no-repeat; background-position:
center;">center</div>
  <div class="box" style="background-image: url('img_chania.jpg');
background-repeat: no-repeat; background-position: bottom
right;">bottom right</div>
</div>
</body>
</html>
```

The preview on the right shows the rendered output. It features a title "background-position examples" and three examples of background images with different positions: "top left", "center", and "bottom right". The "top left" example shows a blue sky image starting from the top-left corner. The "center" example shows a landscape image centered. The "bottom right" example shows a landscape image starting from the bottom-right corner.

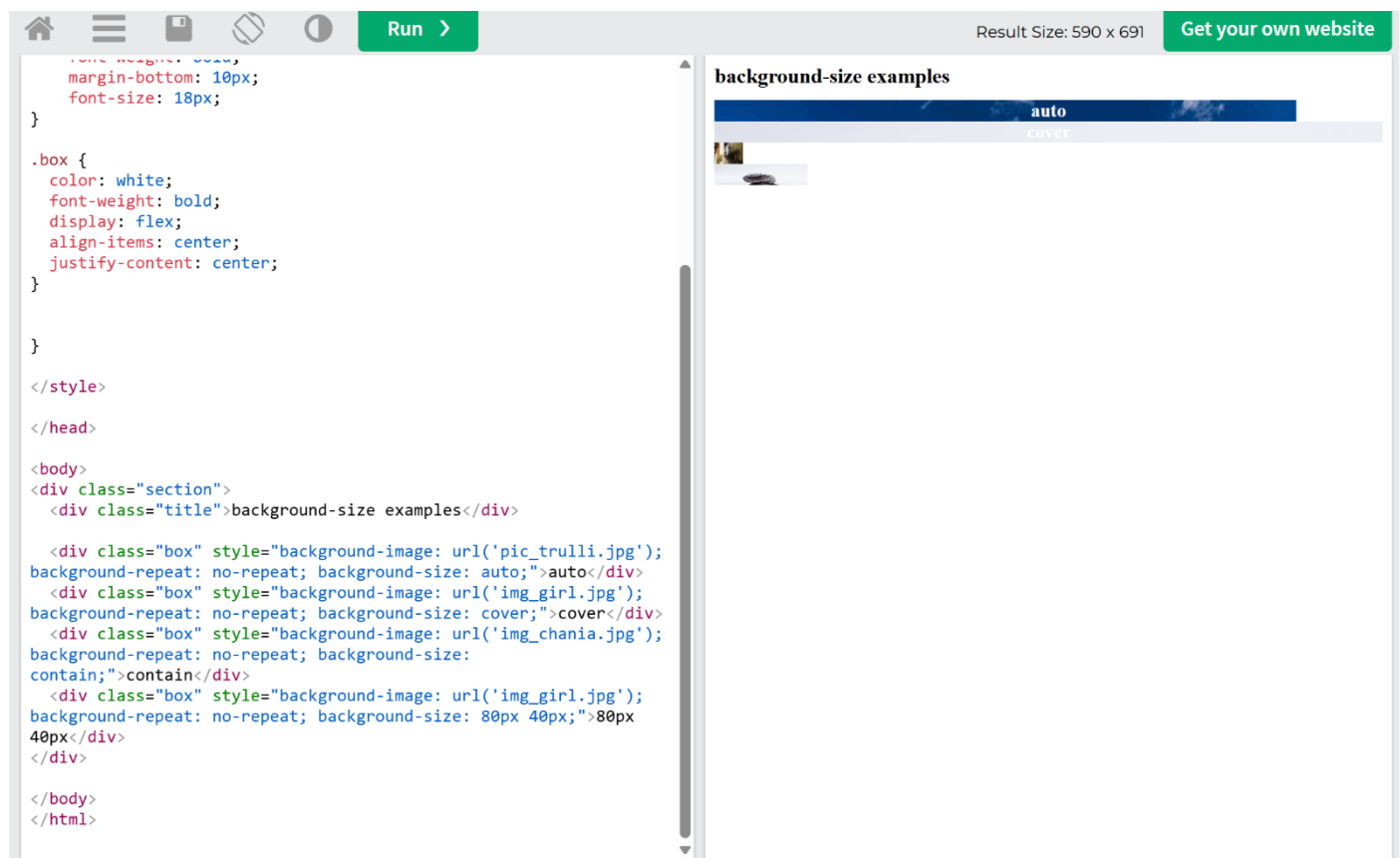
# background-size

## What it does

Controls the **size of the background image**.

## Values

- auto (default) → original size
- cover → fills container, may crop
- contain → fits inside container
- Lengths: 100px 50px



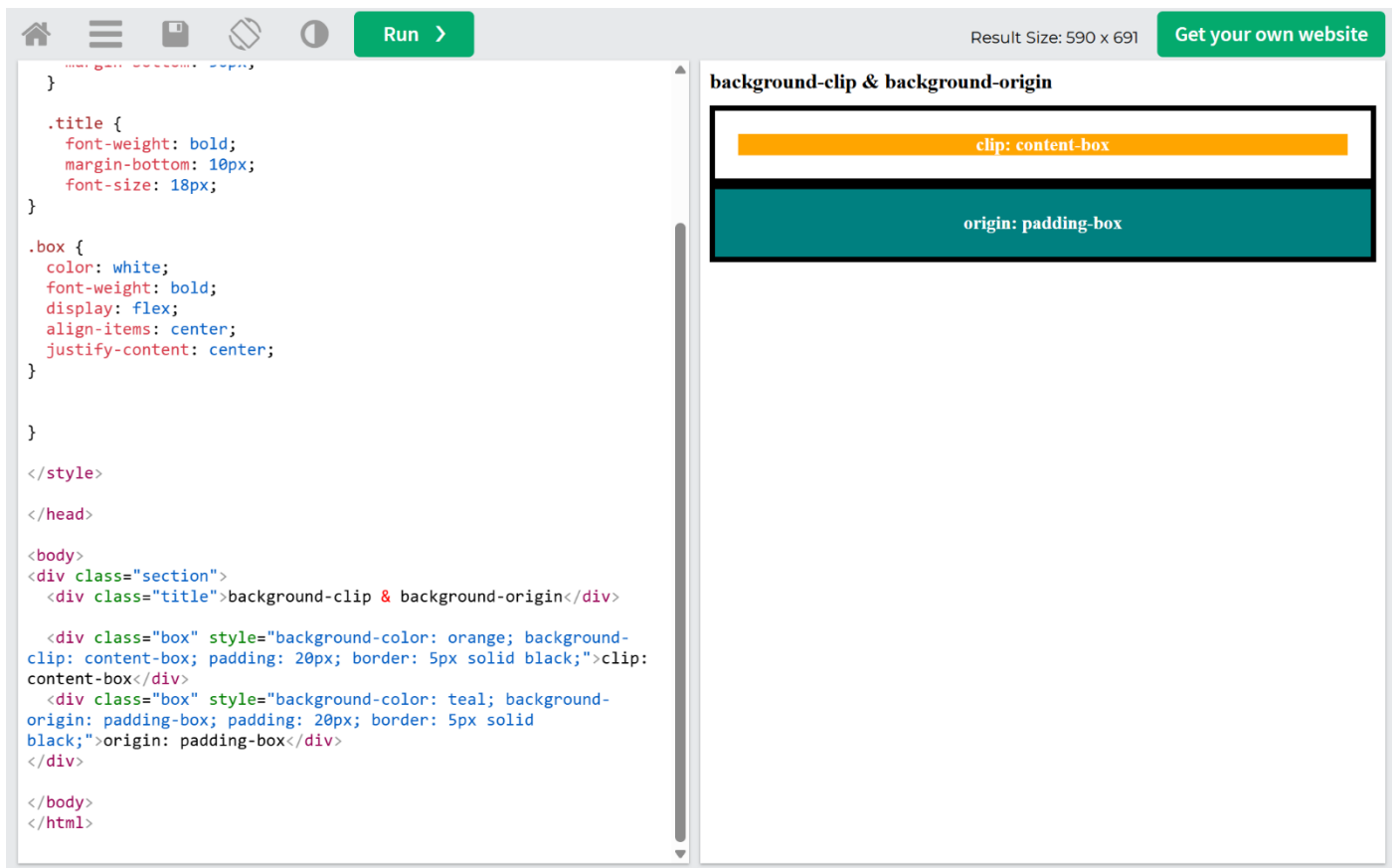
# background-clip & background-origin

## background-clip

- Determines **how far the background extends**
- Values: border-box(default), padding-box(clip outside padding), content-box(clip outside content)

## background-origin

- Determines **where the background image starts**
- Values: border-box(default), padding-box(within border), content-box(from start of content)



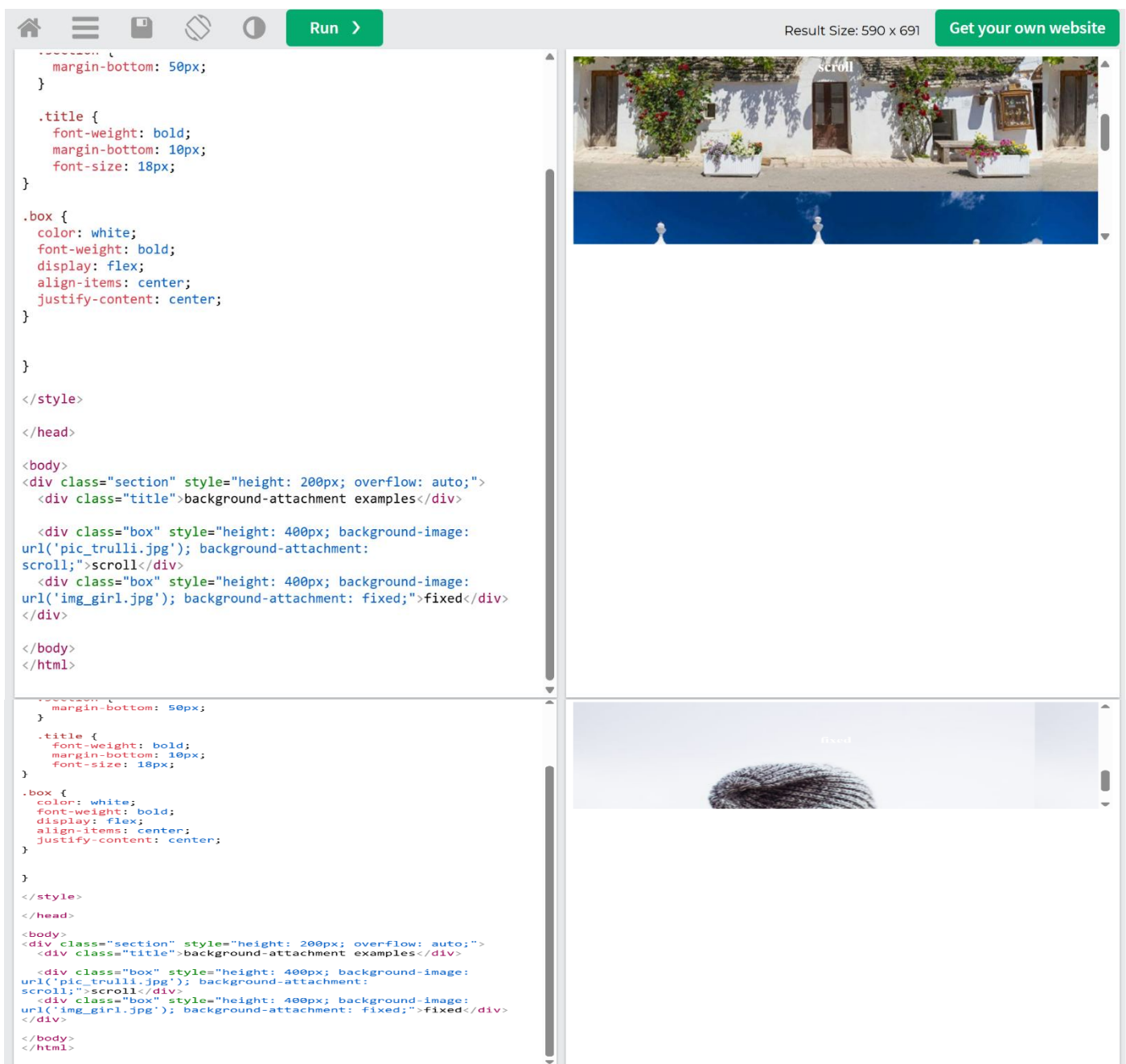
# background-attachment

## What it does

Controls **scrolling behavior** of background image.

## Values

- scroll (default) → moves with content
- fixed → stays in viewport
- local → scrolls with element content



# Overflow

Controls what happens when content exceeds the element's box.

Value	Meaning
visible	Default. Content <b>overflows</b> the box and is fully visible.
hidden	Content is <b>cut off</b> (clipped) at the boundary of the box.
scroll	Always shows <b>scrollbars</b> to access content.
auto	Shows <b>scrollbars only if needed</b> .

**Note:** overflow-x and overflow-y can control horizontal and vertical separately.

The screenshot shows a web browser interface with a code editor on the left and a preview area on the right. The code editor contains HTML snippets for four different overflow settings: hidden, scroll, auto, and visible (default). The preview area, titled "Overflow Examples", shows the corresponding visual results for each setting. The "hidden" example shows text that is completely cut off. The "scroll" example shows text with a vertical scrollbar. The "auto" example shows text with a vertical scrollbar. The "visible (default)" example shows text that overflows the box and is fully visible. The code snippets are as follows:

```
<!-- hidden -->
<div class="box" style="height: 100px; width: 200px; background:
dodgerblue; overflow: hidden; margin-top:10px;">
  Hidden Overflow: Artificial Intelligence (AI) is a field of
computer science that develops systems capable of performing tasks
typically requiring human intelligence, such as reasoning,
learning, problem-solving, and perception.
</div>

<!-- scroll -->
<div class="box" style="height: 100px; width: 200px; background:
seagreen; overflow: scroll; margin-top:10px;">
  Scroll Overflow: Artificial Intelligence (AI) is a field of
computer science that develops systems capable of performing tasks
typically requiring human intelligence, such as reasoning,
learning, problem-solving, and perception.
</div>

<!-- auto -->
<div class="box" style="height: 100px; width: 200px; background:
goldenrod; overflow: auto; margin-top:10px;">
  Auto Overflow: Artificial Intelligence (AI) is a field of
computer science that develops systems capable of performing tasks
typically requiring human intelligence, such as reasoning,
learning, problem-solving, and perception.
</div>

<!-- visible (default) -->
<br><br><br><br><br><br><br><br>
<div class="box" style="color:black;height: 100px; width: 200px;
background: lightcoral; overflow: visible;">
  Visible Overflow: Artificial Intelligence (AI) is a field of
computer science that develops systems capable of performing tasks
typically requiring human intelligence, such as reasoning,
learning, problem-solving, and perception.
</div>
</div>
```

# Filters

Filter	What it does
blur()	Blurs the element using a Gaussian blur
brightness()	Makes the element darker or brighter
contrast()	Increases or decreases difference between light and dark
grayscale()	Converts colors to grayscale
sepia()	Applies a warm brown tone
saturate()	Increases or decreases color intensity
hue-rotate()	Rotates colors around the color wheel
invert()	Inverts colors
opacity()	Controls transparency
drop-shadow()	Adds shadow following the element shape

---

## Important notes

- Filters are applied **left → right**
- Order changes the result
- Filters do **not** affect layout or size


Result Size: 584 x 691 [Get your own website](#)

```
<html>
<head>
<style>
body {
background: #f4f4f4;
font-family: Arial, sans-serif;
}

.demo {
width: 300px;
height: 200px;
margin: 50px auto;
background: linear-gradient(45deg, red, orange, yellow, green,
blue, purple);
display: flex;
align-items: center;
justify-content: center;
font-size: 24px;
color: white;
border-radius: 12px;

filter:
blur(1px)
brightness(120%)
contrast(130%)
grayscale(20%)
sepia(30%)
saturate(150%)
hue-rotate(30deg)
invert(0%)
opacity(90%)
drop-shadow(10px 10px 15px rgba(0,0,0,0.4));
}
</style>
</head>
<body>

<div class="demo">ALL FILTERS</div>
```



# CSS Transforms, Transitions & Animations

This section covers **motion and interaction in CSS**. These properties do **not affect layout flow** — they only affect how elements are visually rendered.

---

## Transforms

Transforms change the **shape, position, size, or rotation** of an element in 2D or 3D space.

### *transform*

Applies one or more transform functions.

---

### *Possible functions*

- `translate()`, `translateX()`, `translateY()`, `translateZ()`
- `scale()`, `scaleX()`, `scaleY()`, `scaleZ()`
- `rotate()`, `rotateX()`, `rotateY()`, `rotateZ()`
- `skew()`, `skewX()`, `skewY()`, `skewZ()`
- `matrix()` takes exactly six parameters, all of which are <number> values: `matrix(a, b, c, d, tx, ty)`. The parameters correspond to specific transformation types:
  - **a and d**: Scaling factors for the X-axis (a) and Y-axis (d).
  - **b and c**: Skew factors for shearing along the Y-axis (b) and X-axis (c).
  - **tx and ty**: Translation (movement) along the X-axis (tx) and Y-axis (ty), effectively in pixels.

Transforms are applied **from right to left**.

---

### *transform-origin*

Defines the point around which the transform occurs.

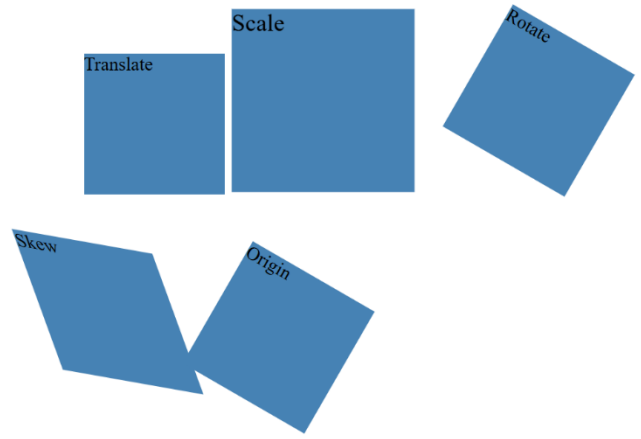
Values:

- center (default)
  - top, bottom, left, right
  - percentages
  - px values
-

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
width: 120px;
height: 120px;
background: steelblue;
margin: 30px;
display: inline-block;
}

.translate { transform: translate(40px, 20px); }
.scale { transform: scale(1.3); }
.rotate { transform: rotate(30deg); }
.skew { transform: skew(20deg, 10deg); }
.matrix{transform: matrix;}

.origin {
transform-origin: top left;
transform: rotate(30deg);
}
</style>
</head>
<body>
<div class="box translate">Translate</div>
<div class="box scale">Scale</div>
<div class="box rotate">Rotate</div>
<div class="box skew">Skew</div>
<div class="box origin">Origin</div>
<div class="box matrix">Matrix</div>
</body>
</html>
```



## 3D Transforms

CSS 3D transforms allow elements to move and rotate in **three-dimensional space**, creating depth and realistic UI effects like card flips and cubes.

---

### *perspective*

Defines the **distance between the viewer and the element** (camera depth).

#### What it does

- Controls strength of 3D depth
- Smaller value → stronger 3D distortion
- Larger value → flatter appearance

#### Values

- none (default)
  - Length values (px only)
- 

### *perspective-origin*

Defines **where the viewer is positioned**.

#### What it does

- Moves the vanishing point
- Changes how depth tilts visually

#### Values

- center (default)
- left | right | top | bottom
- Percentages
- Length values

## *transform-style*

Controls whether child elements **preserve 3D positioning**.

### What it does

- Allows nested elements to stay in 3D space

### Values

- flat (default)
- preserve-3d

---

## *backface-visibility*

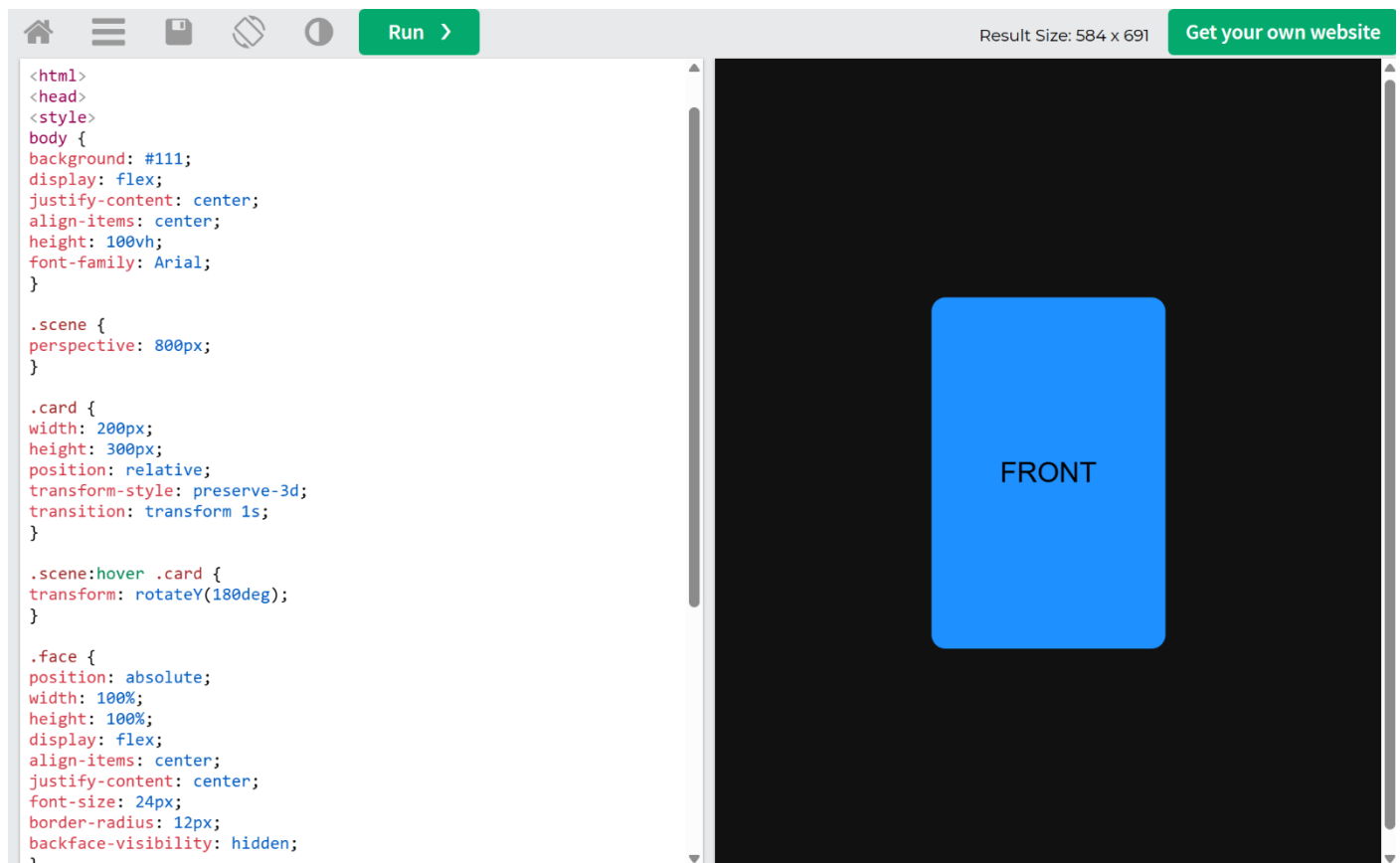
Controls visibility of the **back side** of a rotated element.

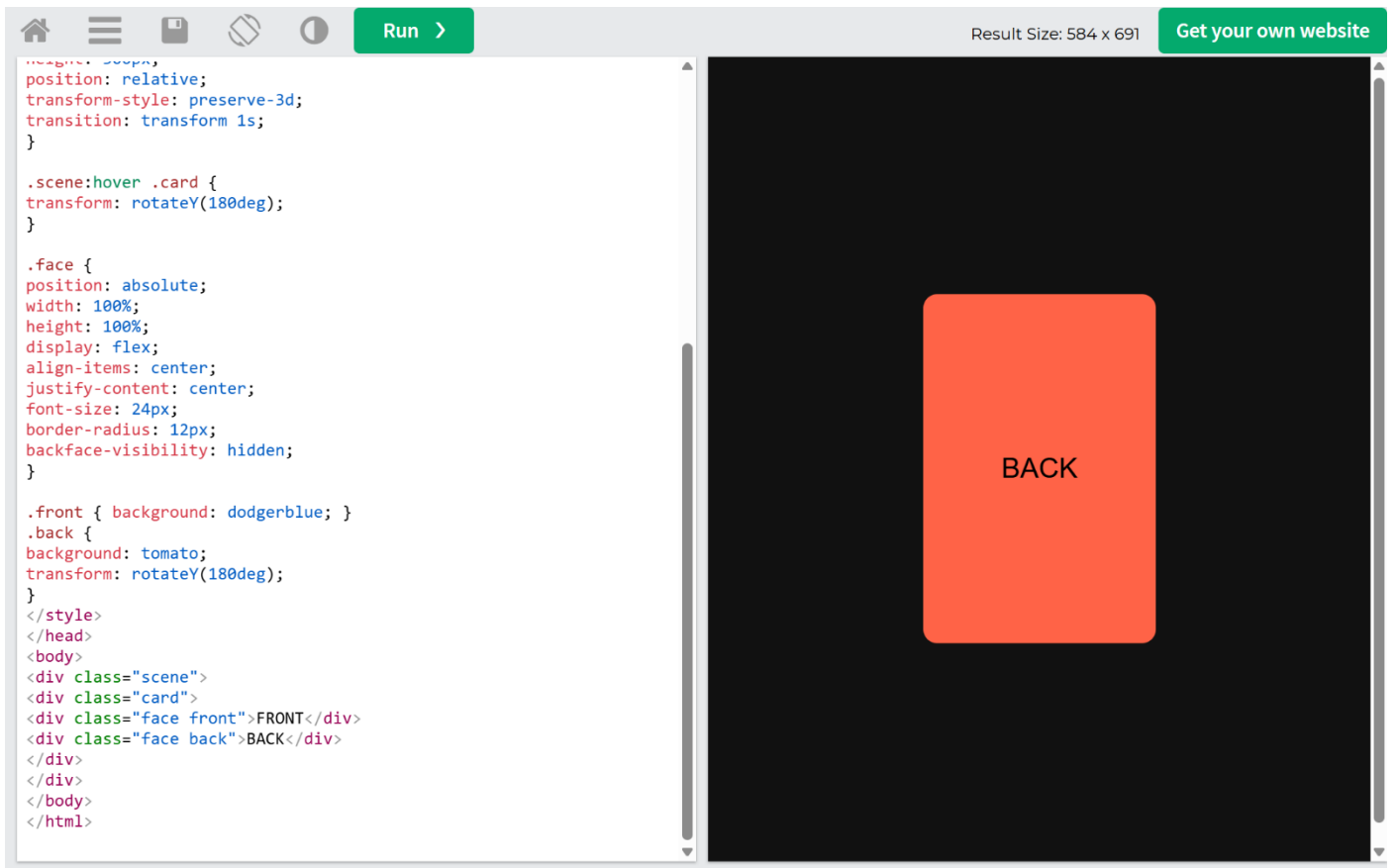
### What it does

- Prevents mirrored text or reversed faces

### Values

- visible (default)
- hidden





## *Transitions*

Transitions animate **property changes over time**.

### *transition*

Shorthand for:

- transition-property
  - transition-duration
  - transition-timing-function
  - transition-delay
- 

### *transition-property*

Which property to animate.

Values:

- all
  - specific property (e.g. width, transform)
- 

### *transition-duration*

How long the transition takes.

Values:

- s
- ms

## *transition-timing-function*

Controls speed curve.

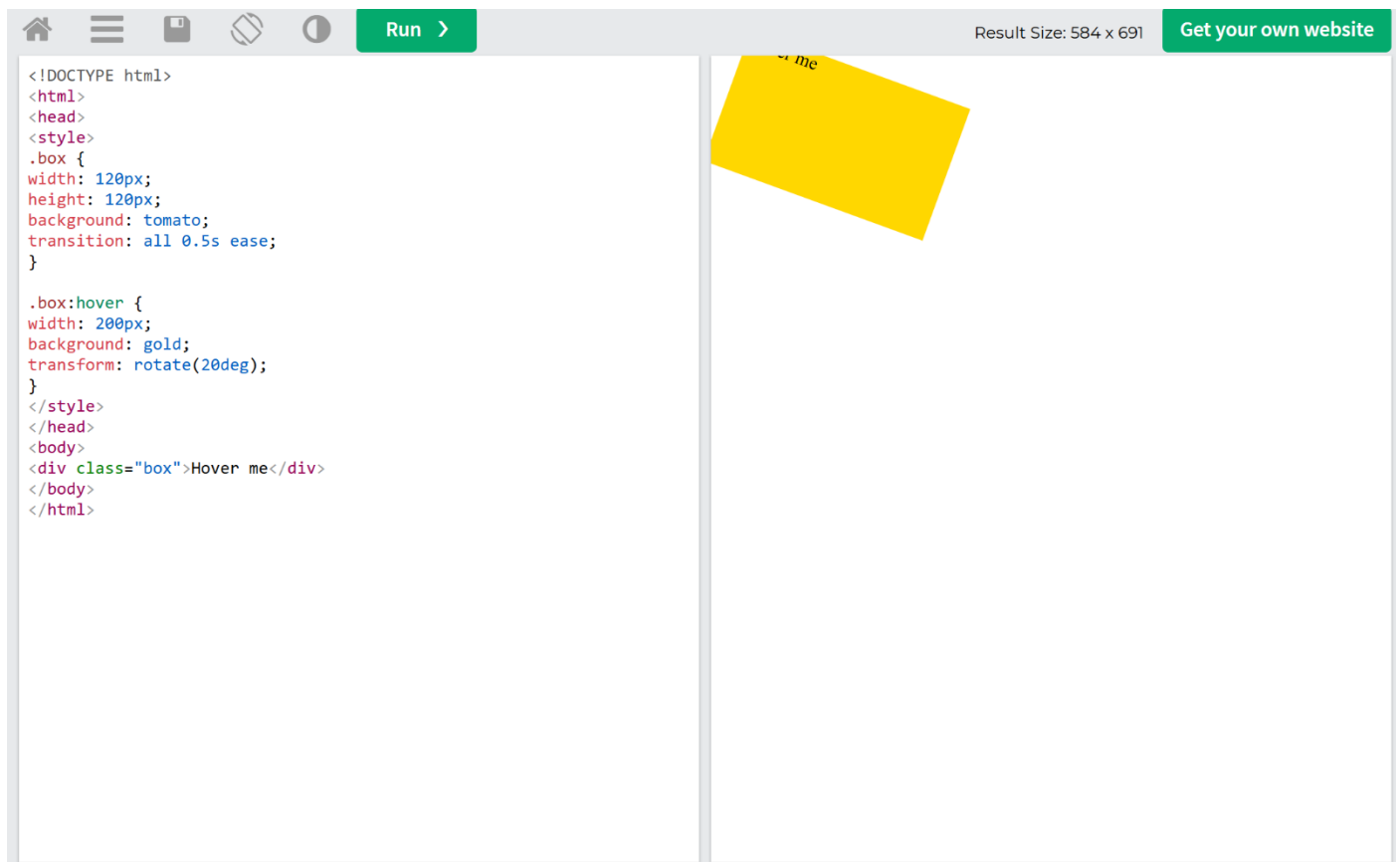
Values:

- ease (default)
- linear
- ease-in
- ease-out
- ease-in-out
- cubic-bezier()

---

## *transition-delay*

Delay before animation starts.



## Animations

Animations allow **automatic motion** using keyframes.

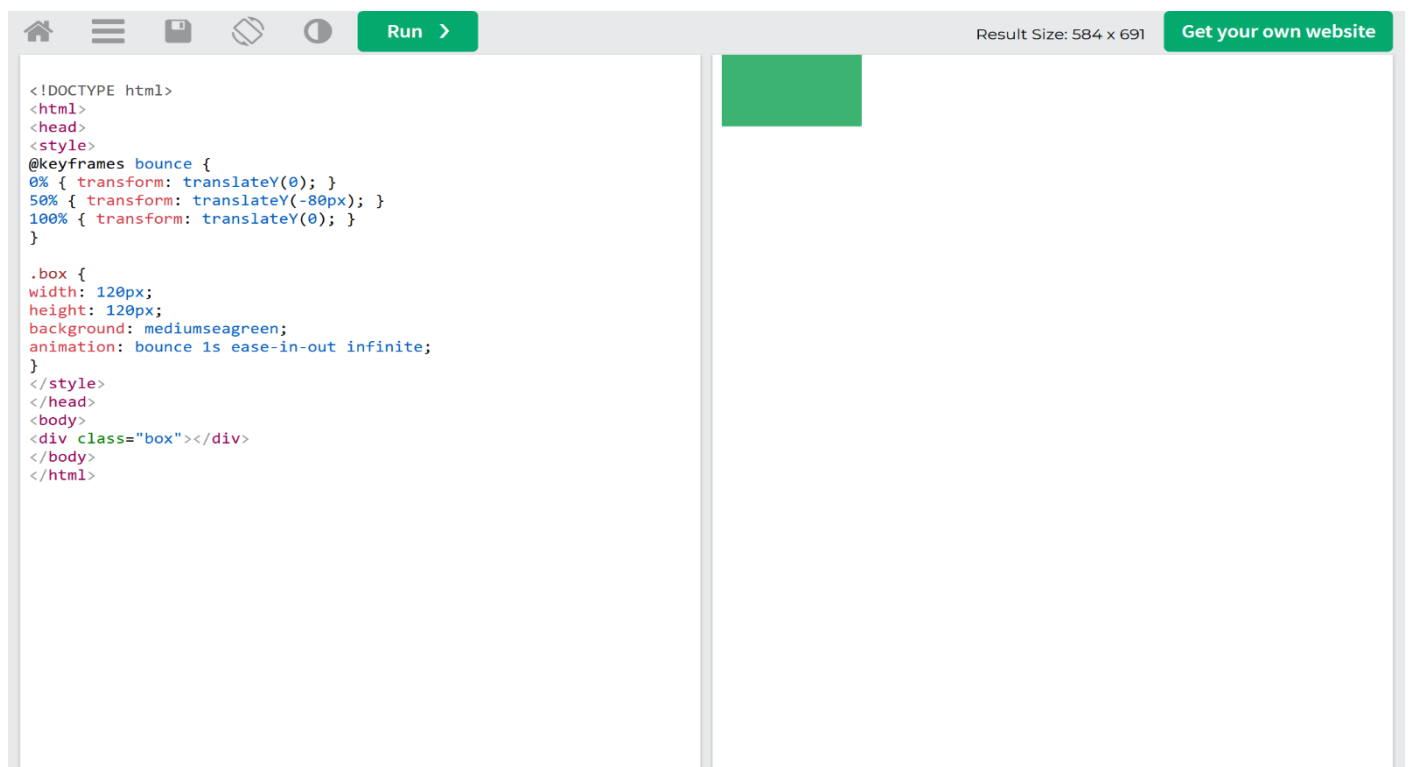
### @keyframes

Defines animation steps.

```
@keyframes move {  
  from { transform: translateX(0); }  
  to { transform: translateX(200px); }  
}
```

### Animation Properties

- **animation-name**
- **animation-duration** (in seconds)
- **animation-timing-function** (ease, linear, ease-in, ease-out, ease-in-out, step-start, step-end, steps(n, start | end, cubic-bezier(x1, y1, x2, y2))
- **animation-delay** (in seconds)
- **animation-iteration-count** (any positive number)
- **animation-direction** (normal, reverse, alternate, alternate-reverse)
- **animation-fill-mode** (none, forwards, backwards, both)
- **animation-play-state** (running, paused)



# Responsive Design

Responsive design means building layouts that **adapt to different screen sizes, devices, and orientations** without breaking usability or readability.

The goal is:

- One website
  - One codebase
  - Works everywhere
- 

## Core Principles of Responsive Design

- Fluid layouts (flexible widths)
  - Flexible images & media
  - Media queries
  - Responsive typography
  - Mobile-first thinking
- 

## Viewport

Controls how the page scales on mobile devices.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Without this, mobile browsers simulate desktop widths.

## Responsive Units

Unit	Description
%	Relative to parent
vw	1% of viewport width
vh	1% of viewport height
vmin	Smaller of vw/vh
vmax	Larger of vw/vh
em	Relative to parent font-size
rem	Relative to root font-size

---

## Responsive Images

```
img {  
  max-width: 100%;  
  height: auto;  
}
```

Prevents images from overflowing containers.

---

## Media Queries

Media queries apply CSS **based on device conditions**.

### Syntax

```
@media (condition) {  
  /* CSS rules */  
}
```

## Breakpoints (Common Practice)

```
/* Mobile */  
@media (max-width: 600px) {}  
  
/* Tablet */  
@media (min-width: 601px) and (max-width: 1024px) {}  
  
/* Desktop */  
@media (min-width: 1025px) {}
```

## Responsive Typography

**clamp(minimum, preferred, maximum)**

Dynamically scales font sizes.

```
h1 {  
font-size: clamp(1.5rem, 4vw, 3rem);  
}
```

## Orientation

```
@media (orientation: landscape) {  
body {  
background: #eee;  
}  
}
```

## Hiding & Showing Content

```
.mobile-only {  
display: block;  
}  
  
.desktop-only {  
display: none;  
}  
  
@media (min-width: 768px) {  
.mobile-only { display: none; }  
.desktop-only { display: block; }  
}
```

## Conclusion

CSS is a powerful tool that enables developers to **create visually appealing and responsive web pages**. Understanding its full range of properties, behaviors, and interactions is essential for mastering web design. By combining **concept explanations, exhaustive value lists, and practical examples**, this document serves as a complete guide for learning and referencing CSS. It empowers developers to **solve layout challenges, style content effectively, and build professional web interfaces** with confidence.

## Feedback & Contribution

This is a living document. If you have feedback, suggestions, or additional insights that could improve it, I welcome your input. Sharing ideas helps keep this material accurate, relevant, and valuable for everyone.

## Copyright & Usage

© 2025 [Youssef Amgad Elkhatib].

This document is intended **for learning and reference purposes only**.

You are free to read, use, and share the content for personal or educational use, but:

- **Do not copy or republish this document as your own.**
- Always provide proper credit when referencing.
- Commercial use or redistribution without permission is not allowed.

You may not reproduce this document in whole or in part without attribution. Suggestions and feedback are welcome, and contributors will be acknowledged, but copyright remains with the author.

# Acknowledgments

This document is authored and copyrighted by **[Youssef Amgad Elkhatib]**.

Special thanks to the contributors who provided valuable feedback and suggestions for improvements:

Name	Contribution