CSS

CSS Backgrounds

- The CSS background properties are used to add background effects for elements.
- We will go through:-
  - background-color
  - background-image
  - background-repeat
  - background-attachment
  - background-position
  - background (shorthand property)

CSS background-attachment

- The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):
- ```
  body {
    background-image: url(“cat.png");
    background-repeat: no-repeat;
    background-position: right top;
    background-attachment: fixed;
  }
  ```
  ```

CSS Background Shorthand

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

When using the shorthand property the order of the property values is:

background-color
background-image
background-repeat
background-attachment
background-position

CSS Fonts

- `font-family: "Times New Roman", Times, serif;`
- `font-family: Arial, Helvetica, sans-serif;`

Note: If the font name is more than one word, it must be in quotation marks, like: "Times New Roman".

What are Web Safe Fonts?

- Web safe fonts are fonts that are universally installed across all browsers and devices.
- there are no 100% completely web safe fonts. There is always a chance that a font is not found or is not installed properly.
- Therefore, it is very important to always use fallback fonts.
- A fallback font is **a font which will be used** if **your other choices aren't usable,** for example in css if we want one font we would write: font-family: font-name;

Best Web Safe Fonts for HTML and CSS

The following list are the best web safe fonts for HTML and CSS:

- Arial (sans-serif)
- Verdana (sans-serif)
- Helvetica (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

Font Fallbacks

Commonly Used Font Fallbacks

Below are some commonly used font fallbacks, organized by the 5 generic font families:

- **Serif**
- **Sans-serif**
- **Monospace**
- **Cursive**
- **Fantasy**

The CSS Font Shorthand

- To shorten the code, it is also possible to specify all the individual font properties in one property.

The font property is a shorthand property for:

- font-style
- font-variant
- font-weight
- font-size/line-height
- font-family

Note: The font-size and font-family values are required. If one of the other values is missing, their default value are used.

```
font: italic small-caps bold 12px/30px Georgia, serif;
```

Text Formatting

- Text Color: `color: blue;`
- Text Color and Background Color:

```
h1 {
  background-color: black;
  color: white;
}
```

Text Alignment and Direction

- text-align : Left | Right | Center | Justify
- text-align-last
- direction
- unicode-bidi
- vertical-align

The text-align-last property specifies how to align the last line of a text.

- `text-align-last: right;` values: Right | Center | Justify

Text Direction

- The direction and unicode-bidi properties can be used to change the text direction of an element:

```
p {
  direction: rtl;
  unicode-bidi: bidi-override;
}
```

Vertical Alignment

- The vertical-align property sets the vertical alignment of an element.

```
img {
  vertical-align: baseline;
}
```

```
Other Values: text-top | text-bottom | sub | super
```

```
<p>Hello There <img src="cat.png" width="9" height="9"> </p>
```

CSS Text Decoration

We will learn about the following properties:

- text-decoration-line
- text-decoration-color
- text-decoration-style
- text-decoration-thickness
- text-decoration

- ```
  h1 {
    text-decoration-line: overline;
  }
  ```

- Other Values : line-through | underline | overline underline
  ```

Specify a Color for the Decoration Line

The text-decoration-color property is used to set the color of the decoration line.

```
h1 {
  text-decoration-line: overline;
  text-decoration-color: red;
}

p {
  text-decoration-line: overline underline;
  text-decoration-color: purple;
}
```

Specify a Style for the Decoration Line

The text-decoration-style property is used to set the style of the decoration line.

```
h1 {
  text-decoration-line: underline;
  text-decoration-style: solid;
}

h2 {
  text-decoration-line: underline;
  text-decoration-style: double;  | dashed | dotted
}

h3 {
  text-decoration-line: underline;
    text-decoration-style: wavy;
}
```

Specify the Thickness for the Decoration Line

The text-decoration-thickness property is used to set the thickness of the decoration line.

```
h1 {
  text-decoration-line: underline;
  text-decoration-thickness: auto;
}
h2 {
  text-decoration-line: underline;
  text-decoration-thickness: 5px | 25%;
}
```

```
p {
  text-decoration-line: underline;
  text-decoration-color: red;
  text-decoration-style: double;
  text-decoration-thickness: 5px;
}
```

The Shorthand Property

The text-decoration property is a shorthand property for:

text-decoration-line (required)

text-decoration-color (optional)

text-decoration-style (optional)

text-decoration-thickness (optional)

```
text-decoration: underline red double 5px;
```

Text Transformation

- The text-transform property is used to specify uppercase and lowercase letters in a text.
- It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:
- ```
  text-transform: uppercase | lowercase | capitalize;
  ```
  ```

Text Spacing

We will learn about the following properties:

text-indent

letter-spacing

line-height

word-spacing

white-space : `white-space: nowrap;`

Text Shadow

- The text-shadow property adds shadow to text.

```
h1 {
  text-shadow: 2px 2px;
    with color
  text-shadow: 2px 2px red;
    with blurness
  text-shadow: 2px 2px 5px red;
    with multiple color
  text-shadow: 0 0 3px #ff0000, 0 0 5px #0000ff;
}
```

CSS Icons

- Icons can easily be added to your HTML page, by using an icon library.

- ```
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  ```

  ```

  ```
  ```

CSS Lists

```
list-style-type: square;
list-style-type: upper-roman;
list-style-type: lower-alpha;
list-style-type: circle;
```

```
ul.a {
  list-style-type: circle;
}
```

```
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

An Image as The List Item Marker

```
ul {
  list-style-image: url(cat.gif');
}
```

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

Position The List Item Markers

The list-style-position property specifies the position of the list-item markers (bullet points).

- list-style-position: outside;" means that the bullet points will be outside the list item. The start of each line of a list item will be aligned vertically. This is default:

- list-style-position: inside;" means that the bullet points will be inside the list item. As it is part of the list item, it will be part of the text and push the text at the start:

Remove Default Settings

- list-style-type:none property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add margin:0 and padding:0 to <ul> or <ol>:

- ```
  ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
  }
  ```
  ```

List - Shorthand property

- list-style: square inside url(“cat.gif");
- When using the shorthand property, the order of the property values are:
- list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
- list-style-position (specifies whether the list-item markers should appear inside or outside the content flow)
- list-style-image (specifies an image as the list item marker)

Styling List With Colors

```
ol {
  background: #ff9999;
  padding: 20px;
}

ul {
  background: #3399ff;
  padding: 20px;
}
```

```
ol li {
  background: #ffe5e5;
  color: darkred;
  padding: 5px;
  margin-left: 35px;
}

ul li {
  background: #cce5ff;
  color: darblue;
  margin: 5px;
}
```

CSS Tables

Table Borders:

To specify table borders in CSS, use the border property.

The example below specifies a solid border for <table>, <th>, and <td> elements:

```
table, th, td {
  border: 1px solid;
}
```

Full Width Table:

```
Table{
        width:100%;
}
```

Collapse Table Borders

- The border-collapse property sets whether the table borders should be collapsed into a single border:

```
table, td, th {
  border: 1px solid;
}

table {
  width: 100%;
  border-collapse: collapse;
}
```

- Outline Only:

```
table {
  width: 100%;
  border: 1px solid;
}
```

Table Alignment

- Horizontal Alignment
- The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.

```
td {
  text-align: center;
}
```

Vertical Alignment

The vertical-align property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

```
table, td, th {
  border: 1px solid black;
}

table {
  border-collapse: collapse;
  width: 100%;
}

td {
  height: 50px;
  vertical-align: bottom;
}
```

CSS Table Style

Table Padding

- To control the space between the border and the content in a table, use the padding property on <td> and <th> elements:

```
th, td {
  padding: 15px;
}
```

Horizontal Dividers

```
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}
```

Hoverable Table

```
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}

tr:hover {background-color: coral;}
```

Striped Tables

For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows:

```
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  text-align: left;
  padding: 8px;
}

tr:nth-child(even) {background-color: #f2f2f2;}
```

CSS Display

- The display property specifies if/how an element is displayed.

- Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

- **What is block level element?**
- A block-level element always starts on a new line and takes up the full width available.

<h1> - <h6>

<p>

<form>

<header>

<footer>

<section>

CSS Display

Inline Elements

- An inline element does not start on a new line and only takes up as much width as necessary.
- This is an inline element inside a paragraph.
- Examples of inline elements:

  <a>

  <img>

Common Display values

- p.{display: none;}
- p.{display: inline;}
- p.{display: block;}
- p.{display: inline-block;}
- p.{display: flex;}
- p.{display: grid;}

- Inline-block: Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values

Disply: Inline-Block Example

```
span {
  display: inline-block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
```

CSS Position

- The position property specifies the type of positioning method used for an element.

- There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

position: static;

- An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

```
div{
  position: static;
  border: 3px solid #73AD21;
}
```

Position: relative;

- An element with position: relative; is positioned relative to its normal position.

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

```
div {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

Position: fixed;

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

```
div {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

Position: absolute;

- An element with position: absolute; is positioned relative to the nearest positioned ancestor.
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

```
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
```

```
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

Position: sticky;

- An element with position: sticky; is positioned based on the user's scroll position.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

```
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
```

Z-index Property

- When elements are positioned, they can overlap other elements.
- The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
- An element can have a positive or negative stack order:

- Tips:-
- Make parent with relative 500px sq. left, top 100px.
- Make 3 Abs Pos inside parent with 350px sq. top, left 10px z-index-1
- Increase left and Right by 50px

CSS Overflow

- The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

- visible - Default. The overflow is not clipped. The content renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
- auto - Similar to scroll, but it adds scrollbars only when necessary

Overflow-x and overflow-y

- The overflow-x and overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both):

- overflow-x specifies what to do with the left/right edges of the content.
- overflow-y specifies what to do with the top/bottom edges of the content.

CSS Navigation Bar

```
<ul>
  <li><a href="default.asp">Home</a></li>
  <li><a href="news.asp">News</a></li>
  <li><a href="contact.asp">Contact</a></li>
  <li><a href="about.asp">About</a></li>
</ul>
```

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

Vertical Navigation Bar

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

li a {
  display: block;
  width: 60px;
  background-color: #dddddd;
}
```

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 200px;
  background-color: #f1f1f1;
}

li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}
```

```
li a.active {
  background-color: #04AA6D;
  color: white;
}

li a:hover:not(.active) {
  background-color: #555;
  color: white;
}
```

Horizontal Navigation Bar

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
}

li {
  float: left;
}
```

```
li a {
  display: block;
  padding: 8px;
  background-color: #dddddd;
}
```

Horizontal Navigation Bar-II

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover {
  background-color: #111;
}
</style>
</head>
<body>
```

Dropdown

```
<div class="dropdown">
  <div class="dropdown-content">
    <p>Hey, Therep>
  </div>
</div>
```

```
.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover .dropdown-content {
  display: block;
}
```

Dropdown Menu

```
<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
  <a href="#">Link 1</a>
  <a href="#">Link 2</a>
  <a href="#">Link 3</a>
  </div>
</div>
```

P.T.O...

Dropdown Menu (Styling)

```
.dropbtn {
  background-color: #4CAF50;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
  cursor: pointer;
}
```

```
.dropdown {
  position: relative;
  display: inline-block;
}
```

```
.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}
```

```
.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}

.dropdown-content a:hover {background-color: red}
```

```
.dropdown:hover .dropdown-content {
  display: block;
}
```

```
.dropdown:hover .dropbtn {
  background-color: #3e8e41;
}
```

CSS Forms

Styling Input Fields

input[type=text] - will only select text fields

input[type=password] - will only select password fields

input[type=number] - will only select number fields

Padded Inputs

```
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
}
```

Use the padding property to add space inside the text field.

Tip: When you have many inputs after each other, you might also want to add some margin, to add more space outside of them:

Bordered Inputs

Use the border property to change the border size and color, and use the border-radius property to add rounded corners:

```
input[type=text] {
  border: 2px solid red;
  border-radius: 4px;
}
```

If you only want a bottom border, use the border-bottom property:

```
input[type=text] {
  border: none;
  border-bottom: 2px solid red;
}
```

Colored Inputs

- Use the background-color property to add a background color to the input, and the color property to change the text color:

```
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
  border: none;
  background-color: #3CBC8D;
  color: white;
}
```

Focused Inputs

- By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input.

```
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
  border: 1px solid #555;
  outline: none;
}

input[type=text]:focus {
  background-color: lightblue;
}
```

Input with icon/image

If you want an icon inside the input, use the background-image property and position it with the background-position property.

```
input[type=text] {
  width: 100%;
  box-sizing: border-box;
  border: 2px solid #ccc;
  border-radius: 4px;
  font-size: 16px;
  background-color: white;
  background-image: url('searchicon.png');
  background-position: 10px 10px;
  background-repeat: no-repeat;
  padding: 12px 20px 12px 40px;
}
```

Animated Search Input

we use the CSS transition property to animate the width of the search input when it gets focus.

```
input[type=text] {
  width: 130px;
  box-sizing: border-box;
  border: 2px solid #ccc;
  border-radius: 4px;
  font-size: 16px;
  background-color: white;
  background-image: url('searchicon.png');
  background-position: 10px 10px;
  background-repeat: no-repeat;
  padding: 12px 20px 12px 40px;
  transition: width 0.4s ease-in-out;
}
```

```
input[type=text]:focus {
  width: 100%;
}
```

Styling Textareas

Use the resize property to prevent textareas from being resized (disable the "grabber")

```
textarea {
  width: 100%;
  height: 150px;
  padding: 12px 20px;
  box-sizing: border-box;
  border: 2px solid #ccc;
  border-radius: 4px;
  background-color: #f8f8f8;
  font-size: 16px;
  resize: none;
}
```

Styling Select Menus

```
select {
  width: 100%;
  padding: 16px 20px;
  border: none;
  border-radius: 4px;
  background-color: #f1f1f1;
}
```

Styling Input Buttons

```
input[type=button], input[type=submit], input[type=reset] {
  background-color: #04AA6D;
  border: none;
  color: white;
  padding: 16px 32px;
  text-decoration: none;
  margin: 4px 2px;
  cursor: pointer;
}
```

CSS Writing Mode

- The CSS writing-mode property specifies whether lines of text are laid out horizontally or vertically.
- Writing-mode: horizontal-tb/ vertical-rl
- Some text with a span element with a vertical-rl writing-mode.

CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

translate()

- The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

```
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
  transform: translate(50px,100px);
}
```

rotate() Method

- The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

```
div {
  width: 300px;
  height: 100px;
  background-color: red;
  border: 1px solid black;
}

div#myDiv {
  transform: rotate(20deg);
}
```

scale() Method

- The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).
- Two times of its original width and 3times of height:

```
div {
  margin: 150px;
  width: 200px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
  transform: scale(2,3); or transform: scale(0.5, 0.5);
}
```

The scaleX() & scaleY() Method

- The scaleX() method increases or decreases the width of an element.

- Two times of its original width:

```
transform: scaleX(2);
```

- The scaleY() method increases or decreases the height of an element.

```
transform: scaleY(3);
```

The skewX() Method

- The skewX() method skews an element along the X-axis by the given angle.

```
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}

div#myDiv {
  transform: skewX(20deg);
}
```

The skewX() Method

- skews the element 20 degrees along the X-axis, and 10 degrees along the Y-axis.

- If the second parameter is not specified, it has a zero value.

CSS 3D Transforms

- With the CSS transform property you can use the following 3D transformation methods:

rotateX()

rotateY()

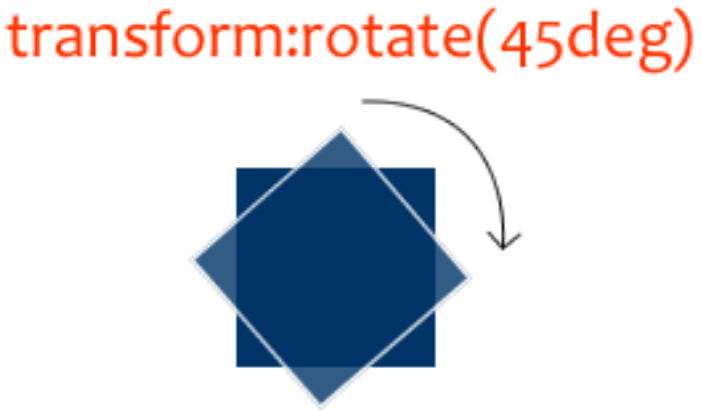
rotateZ()

rotateX()

The rotateX() method rotates an element around its X-axis at a given degree:

```
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}

#myDiv {
  transform: rotateX(150deg);
}
```



rotateY()

- The rotateY() method rotates an element around its Y-axis at a given degree:

```
#myDiv {
  transform: rotateY(150deg);
}
```

rotateZ() Method

- The rotateZ() method rotates an element around its Z-axis at a given degree:

```
#myDiv {
  transform: rotateZ(90deg);
}
```

CSS Transitions

- CSS transitions allows you to change property values smoothly, over a given duration.

transition

transition-delay

transition-duration

transition-property

transition-timing-function

To use CSS Transitions?

You must specify two things:

the CSS property you want to add an effect to

the duration of the effect.

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

Transition effect for the width property, with a duration of 2 seconds:

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}

div:hover {
  width: 300px;
}
```

Adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s, height 4s;
}

div:hover {
  width: 300px;
  height: 300px;
}
```

Specify the Speed Curve of the Transition

- The transition-timing-function property specifies the speed curve of the transition effect.

Values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end

transition-timing property

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}

div1 {transition-timing-function: linear;}

div:hover {
  width: 300px;
}
```

transition-delay

- The transition-delay property specifies a delay (in seconds) for the transition effect.

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 3s;
  transition-delay: 1s;
}

div:hover {
  width: 300px;
}
```

Transition + Transform

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s, height 2s, transform 2s;
}

div:hover {
  width: 300px;
  height: 300px;
  transform: rotate(180deg);
}
```

transition & transition Shorthand Property

```
div {
  transition-property: width;
  transition-duration: 2s;
  transition-timing-function: linear;
  transition-delay: 1s;
}
```

```
div:hover {
  width: 300px;
}
```

Or

```
div {
  transition: width 2s linear 1s;
}
```

CSS Animations

- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

@keyframes Rule

- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.
- To get an animation to work, you must bind the animation to an element.

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: hello;
  animation-duration: 4s;
}
```

```
@keyframes hello {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

```
@keyframes hello {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}
```

```
@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
```

Tootip Arrow

```
.tooltip {
  position: relative;
  display: inline-block;
  border-bottom: 1px dotted black;
}

.tooltip:hover .tooltiptext {
  visibility: visible;
}

<div class="tooltip">Hover over me
  <span class="tooltiptext">Tooltip text</span>
</div>
```

```
.tooltip .tooltiptext {
  visibility: hidden;
  width: 120px;
  background-color: black;
  color: #fff;
  text-align: center;
  border-radius: 6px;
  padding: 5px 0;
  position: absolute;
  z-index: 1;
  bottom: 150%;
  left: 50%;
  margin-left: -60px;
}
```

```
.tooltip .tooltiptext::after {
  content: "";
  position: absolute;
  top: 100%;
  left: 50%;
  margin-left: -5px;
  border-width: 5px;
  border-style: solid;
  border-color: black transparent transparent transparent;
}
```

Image Reflections

- The box-reflect property is used to create an image reflection.

```
img {
  -webkit-box-reflect: right;
}
```

Or

```
-webkit-box-reflect: below 0px linear-gradient(to bottom, rgba(0,0,0,0.0), rgba(0,0,0,0.4));
```

Animation Direction

- The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles.

- The animation-direction property can have the following values:

normal - The animation is played as normal (forwards). This is default

reverse - The animation is played in reverse direction (backwards)

alternate - The animation is played forwards first, then backwards

alternate-reverse - The animation is played backwards first, then forwards

Understanding more about animation.

```
div {
  width: 100px;
  height: 50px;
  background-color: red;
  font-weight: bold;
  position: relative;
  animation: mymove 5s infinite;
}

#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}

@keyframes mymove {
  from {left: 0px;}
  to {left: 300px;}
}
```

```
<div id="div1">linear</div>
<div id="div2">ease</div>
<div id="div3">ease-in</div>
<div id="div4">ease-out</div>
<div id="div5">ease-in-out</div>
```

Animation Fill Mode

```
Div{
    width:150px;
    Height:150px;
    Background:black;
    Margin-left:20px;
    Animation: example 2s 5s;
    Animation-fill-mode: backward(starts by 0% keyframe property) | forwards (won't come to its initial state at the end)| both
}

@keyframes sample {
    0%{background:red; width:250;}
    50%{background:blue; width:250;}
    100%{background:green; width:250;}
}
```

CSS Variables

- The var() function is used to insert the value of a CSS variable.
- CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavaScript, and change the variables based on media queries.

- **Advantages of using var() are:**

  makes the code easier to read (more understandable)

  makes it much easier to change the color values
- To change the blue and white color to a softer blue and white, you just need to change the two variable values:

```
:root {
  --blue: #6495ed;
  --white: #faf0e6;
}

body {
  background-color: var(--blue);
}

h2 {
  border-bottom: 2px solid var(--blue);
}

.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px;
}

button {
  background-color: var(--white);
  color: var(--blue);
  border: 1px solid var(--blue);
  padding: 5px;
}
```

```
<div class="container">
  <h2>Lorem Ipsum</h2>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.
</p>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.
</p>
  <p>
    <button>Yes</button>
    <button>No</button>
  </p>
</div>
```

CSS Flexbox

Before the Flexbox Layout module, there were four layout modes:

- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data
- Positioned, for explicit position of an element.

- The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

Flexbox Example

```
.flex-container {
  display: flex;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
```

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

The flex container properties:

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

flex-direction

- The flex-direction property defines in which direction the container wants to stack the flex items.

- The **column** value stacks the flex items vertically (from top to bottom):
- The **column-reverse** value stacks the flex items vertically (but from bottom to top):
- The **row** value stacks the flex items horizontally (from left to right):
- The **row-reverse** value stacks the flex items horizontally (but from right to left):

flex-direction (Example)

```
.flex-container {
  display: flex;
  flex-direction: column;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

flex-wrap Property

The flex-wrap property specifies whether the flex items should wrap or not.

- The wrap value specifies that the flex items will wrap if necessary:
- The nowrap value specifies that the flex items will not wrap (this is default):
- The wrap-reverse value specifies that the flexible items will wrap if necessary, in reverse order:

flex-wrap example

```
.flex-container {
  display: flex;
  flex-wrap: wrap;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
  <div>11</div>
  <div>12</div>
</div>
```

Flexbox justify-content Property:

The justify-content property is used to align the flex items:

- The center value aligns the flex items at the center of the container:
- The flex-start value aligns the flex items at the beginning of the container (this is default):
- The flex-end value aligns the flex items at the end of the container:
- The flex-end value aligns the flex items at the end of the container:
- The space-between value displays the flex items with space between the lines:

justify-content prop example

```
.flex-container {
  display: flex;
  justify-content: space-around;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

Perfect Centering Divs

```
.flex-container {
  display: flex;
  height: 300px;
  justify-content: center;
  align-items: center;
}
```

CSS clip-path Property

- The clip-path property lets you clip an element to a basic shape or to an SVG source.

```
Img{
Clip-path:circle (40% at 50% 50%);
Clip-path:ellipse (10% 40% at 50% 50%);
Clip-path:inset (5% 20% 15% 10%);
Clip-path: polygon ();
}
```