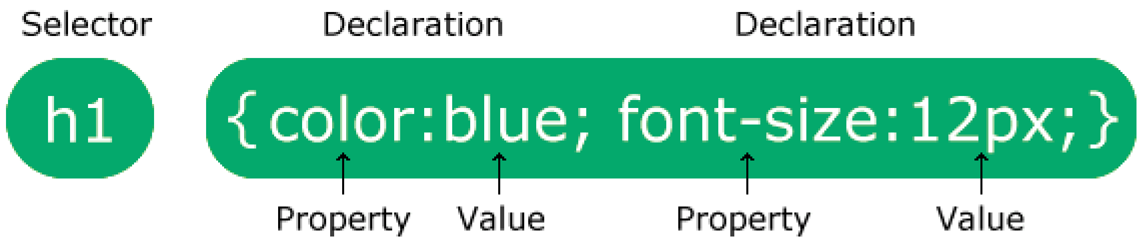
What is CSS

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Syntax



How to add CSS

1. Inline CSS
2. Internal CSS
3. External CSS

1) Inline CSS

Inline CSS is used to apply CSS on a single line or element.

For example:

```
<p style="color:blue">Hello CSS</p>
```

2) Internal CSS

Internal CSS is used to apply CSS on a single document or page. It can affect all the elements of the page. It is written inside the style tag within the head section of html.

For example:

```
<style>
p{color:blue}
</style>
```

3) External CSS

External CSS is used to apply CSS on multiple pages or all pages. Here, we write all the CSS code in a css file. Its extension must be .css **for example style.css.**

For example:

```
p{color:blue}
```

You need to link this style.css file to your html pages like this:

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

The link tag must be used inside head section of html.

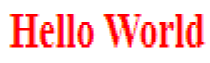
CSS Comments

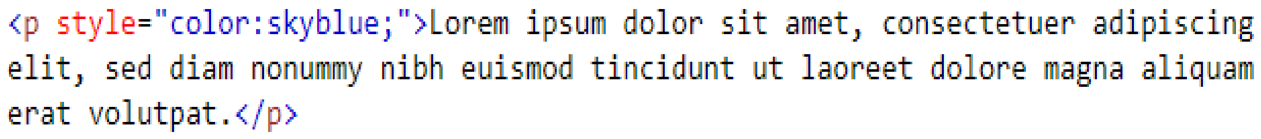
- CSS comments are generally written to explain your code.
- It is very helpful for the users who read your code so that they can easily understand the code.
- Comments are ignored by browsers.
- Comments are single or multiple lines statement and written within /*...........*/ .

CSS Colors

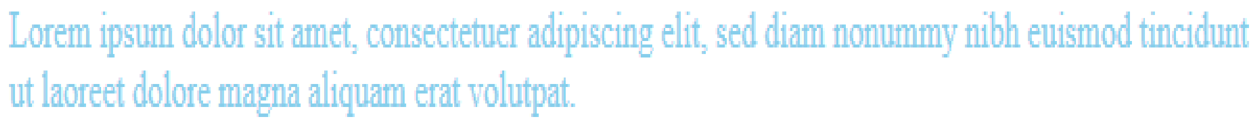
```
<h3 style="color:red;">Hello World</h3>
```

→ Output on Browser


Hello World


```
<p style="color:skyblue;">Lorem ipsum dolor sit amet, consectetuer adipiscing
elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.</p>
```

→ Output on Browser


Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

CSS Color Values

In CSS, colors can also be specified using

1. **RGB** values,
2. **HEX** values,
3. **HSL** values,
4. **RGBA** values, and
5. **HSLA** values.

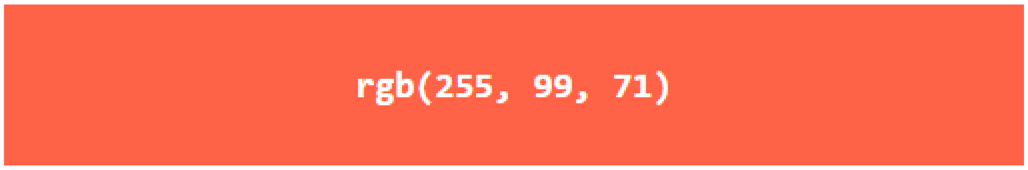
CSS RGB Colors

An RGB color value represents **RED**, **GREEN**, and **BLUE** light sources.

RGB Valueconsole.log(n[i] + " is " + ++a);

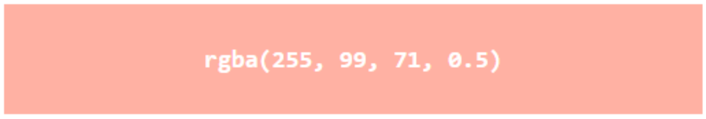
In CSS, a color can be specified as an RGB value, using this formula:

rgb(*red*, *green*, *blue*)


rgb(255, 99, 71)

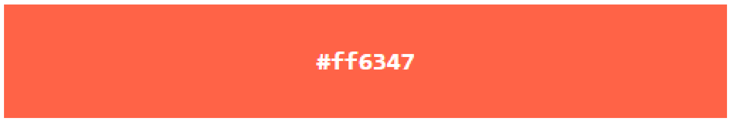
RGBA Value

- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- An RGBA color value is specified with:
- **rgba(*red*, *green*, *blue*, *alpha*)**
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all).


rgba(255, 99, 71, 0.5)

HEX Value

A hexadecimal color is specified with: **#RRGGBB**, where the **RR** (red), **GG** (green) and **BB** (blue) hexadecimal integers specify the components of the color.


#ff6347

HSL Value

In CSS, a color can be specified using **hue**, **saturation**, and **lightness** (HSL) in the form:

hsl(*hue, saturation, lightness*)

hsl(0, 100%, 50%)

- Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
- Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.
- Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white.

HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

hsla(*hue, saturation, lightness, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

hsla(0, 100%, 50%, 0.5)

<u>**CSS Backgrounds**</u>

1. `background-color`
2. `Background-image`
3. `background-size`
4. `Background-repeat`
5. `Background-position`
6. `Background-attachment`
7. `background` (shorthand property)

1. **`background-color`** (use for background color of an element)

    Example:- `{background-color: lightblue;}`

2. **`background-image`** (use for background image of an element)

    Example:

    - Single Background image:
    `{background-image:url("paper.gif");}`
    - Multiple Background image:
    `{background-image:url("paper.gif"),url("img.gif");}`

    ➢ By default, the `background-image` property repeats an image both horizontally and vertically.
    ➢ Some images should be repeated only horizontally or vertically, or they will look strange.

3. **`background-size`** (use for background size of an element)
    ➢ **background-size:** auto | length | cover | contain | initial ;

    Example:
    - For Single Background image:
    `{background-size:width height;}`
    - For Multiple Background image:

```
{background-image:
     1st-bg-width 1st-bg-height,
     2nd-bg-width 2nd-bg-height;}
```

4. **`background-repeat`** (use for handle background repeats)

- ➢ By default, the image is repeated so it covers the entire element.
- ➢ **background-repeat:** repeat|repeat-x|repeat-y|no-repeat;

Example:

- For Single Background image:

`{background-repeat:no-repeat;}`

- For Multiple Background image:

`{background-repeat:no-repeat, repeat-x;}`

5. **`background-position`** (use to set background position)
   - ➢ **background-position:** left top | left center | left bottom | right top | right center | right bottom | center top | center center | center bottom;
   - ➢ If you only specify one keyword, the other value will be "center"
   - ➢ The first value is the horizontal position and the second value is the vertical position.

Example:

- For Single Background image:

`{background-position:left top;}`

- For Multiple Background image:

`{background-position:30px 40px , right bottom;}`

6. **`background-attachment`** (property specifies whether the background image should scroll or be fixed)
   - ➢ **background-attachment:** fixed | scroll | local;
   - ➢ If you only specify one keyword, the other value will be "center"

Example:

- For Single Background image:

`{background-attachment:scroll;}`

- For Multiple Background image:

`{background-attachment:fixed, scroll;}`

7. **`background`** (background Shorthand property)

Example:

```
{background:color image repeat attachment position;}
```

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

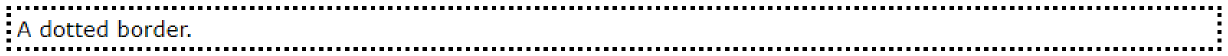
CSS Borders

The CSS border properties allow you to specify the style, width, and color of an element's border.

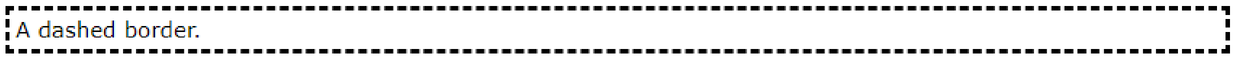
CSS border-style

- ➢ The `border-style` property specifies what kind of border to display.
- ➢ By default the border style color is black.
- ➢ Border-style: `dotted | dashed | solid | double | groove | ridge | inset | outset | none | hidden`

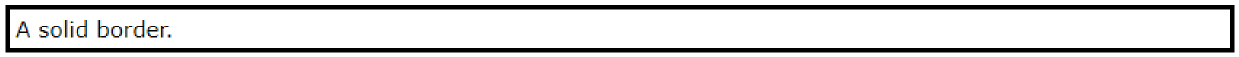
Example: `{border-style: dotted;}`


A dotted border.

Example: `{border-style: dashed;}`


A dashed border.

Example: `{border-style: solid;}`


A solid border.

Example: `{border-style: double;}`

A double border.

Example: `{border-style: groove;}`

A groove border. The effect depends on the border-color value.

Example: `{border-style: ridge;}`

A ridge border. The effect depends on the border-color value.

Example: `{border-style: inset;}`

An inset border. The effect depends on the border-color value.

Example: `{border-style: outset;}`

An outset border. The effect depends on the border-color value.

Example: `{border-style: none;}`

No border.

Example: `{border-style: hidden;}`

A hidden border.

Example: `{border-style: dotted dashed solid double;}`

A mixed border.

CSS Border Width

The `border-width` property specifies the width of the four borders.

Example: `{border-width: 5px;}`

CSS Border Color

The `border-color` property specifies the width of the four borders.

Example: `{border-color: red;}`

CSS Border - Shorthand Property

The `border` property is a shorthand property for the following individual border properties:

- `border-width`
- `border-style` (required)
- `border-color`

Example: `{ border: 5px solid red;}`

CSS Margins

The CSS `margin` properties are used to create space around elements, outside of any defined borders.

This element has a margin of 70px.

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

- `margin-top`
- `margin-right`
- `margin-bottom`
- `margin-left`

Margin - Shorthand Property

If the `margin` property has **four** values:

- margin: 25px 50px 75px 100px;
    - top margin is 25px
    - right margin is 50px
    - bottom margin is 75px
    - left margin is 100px

If the `margin` property has **three** values:

- margin: 25px 50px 75px;
    - top margin is 25px
    - right and left margin are 50px
    - bottom margin is 75px

If the `margin` property has **two** values:

margin: 25px 50px;

- top and bottom margin are 25px
- right and left margin are 50px

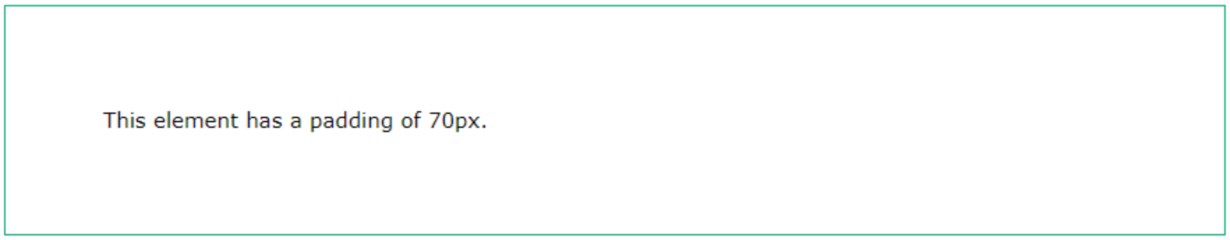
If the `margin` property has **one** value:

margin: 25px;

- top , right, bottom and left margin are 25px

CSS Padding

Padding is used to create space around an element's content, inside of any defined borders.


This element has a padding of 70px.

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `Padding-left`

Padding - Shorthand Property

If the `padding` property has **four** values:

- padding: 25px 50px 75px 100px;
    - top padding is 25px
    - right padding is 50px
    - bottom padding is 75px
    - left padding is 100px

If the `padding` property has **three** values:

- padding: 25px 50px 75px;
    - top padding is 25px
    - right and left paddings are 50px
    - bottom padding is 75px

If the `padding` property has **two** values:

padding: 25px 50px;

- top and bottom paddings are 25px
- right and left paddings are 50px

If the `padding` property has **one** value:

padding: 25px;

- top , right, bottom and left padding are 25px

CSS Height, Width and Max-width

The CSS `height` and `width` properties are used to set the height and width of an element.

The CSS `max-width` property is used to set the maximum width of an element.

The CSS `min-width` property is used to set the minimum width of an element.

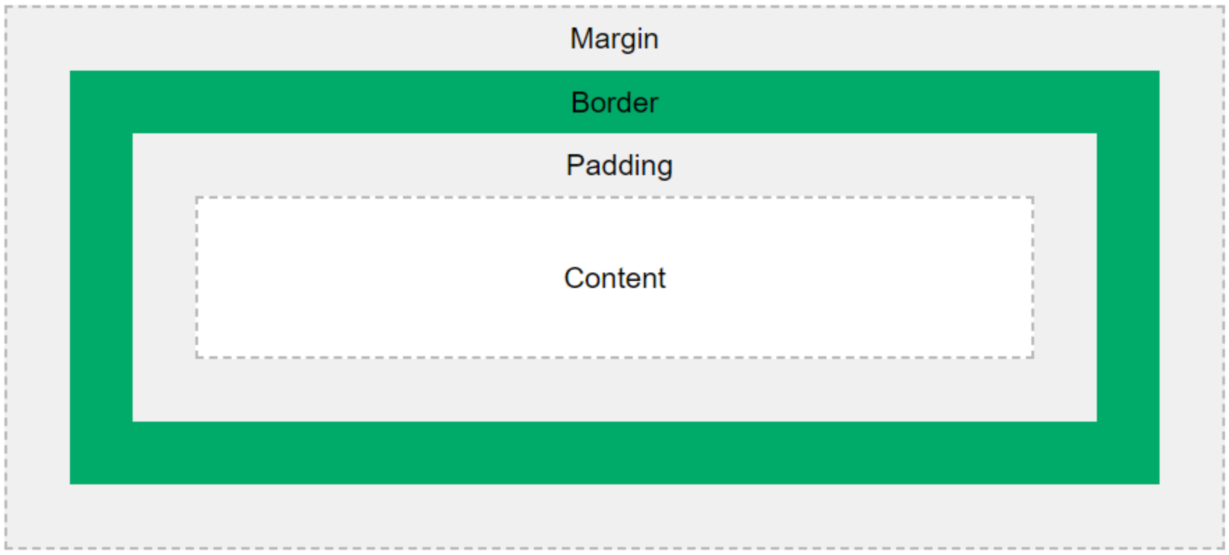
CSS Setting height and width

The `height` and `width` properties are used to set the height and width of an element.

CSS Box Model

In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

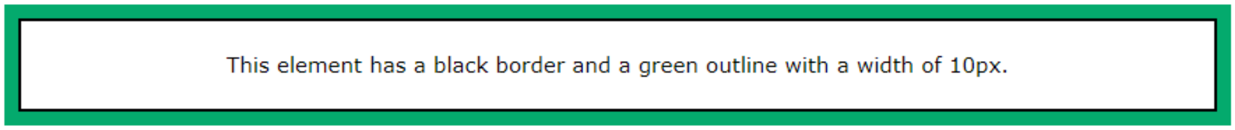


Explanation of the different parts:

- Content - The content of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent

CSS Outline

An outline is a line drawn outside the element's border.


This element has a black border and a green outline with a width of 10px.

CSS has the following outline properties:

- `outline-style`
- `outline-color`
- `outline-width`
- `outline-offset`(it used to make distance between the border and outline)
- `outline`**(1px solid red)**

CSS Outline Style

The `outline-style` property specifies the style of the outline, and can have one of the following values:

- `dotted` - Defines a dotted outline
- `dashed` - Defines a dashed outline
- `solid` - Defines a solid outline
- `double` - Defines a double outline
- `groove` - Defines a 3D grooved outline
- `ridge` - Defines a 3D ridged outline
- `inset` - Defines a 3D inset outline
- `outset` - Defines a 3D outset outline
- `none` - Defines no outline
- `hidden` - Defines a hidden outline

CSS Outline - Shorthand property

The `outline` property is a shorthand property for setting the following individual outline properties:

- `outline-width`
- `outline-style` (required)
- `outline-color`

```
outline: 1px solid red;
```

CSS Text

CSS has a lot of properties for formatting text.

CSS Text Alignment

In this chapter you will learn about the following properties:

- `text-align`

  ```
  text-align: left|right|center|justify|initial|inherit;
  ```

- `Text-align-last` **(it aligns the last line of paragraph and justify sets the content as equal space)**

  ```
  text-align-last:auto|left|right|center|justify|start|end|initia
  l|inherit;
  ```

- `Direction` **(it decides the direction of text alignment)**

  ```
  direction: ltr|rtl|initial|inherit;
  ```

- `Unicode-bidi` **(it will change the text written direction (only bidi-override))**

  ```
  unicode-bidi: normal|embed|bidi-override|initial|inherit;
  ```

`Vertical-align`

```
vertical-align:
baseline(default)|sub|super|top|text-top|middle|bottom|text-bottom|in
itial|inherit;
```

CSS Text Decoration

In this chapter you will learn about the following properties:

- **`text-decoration-line`**

- **`text-decoration-color`**

  `text-decoration-color: *color*|initial|inherit;`

- **`text-decoration-style`**

  `text-decoration-style:` solid|double|dotted|dashed|wavy|initial|inherit;

- **`text-decoration-thickness`**

  `Text-decoration-thickness:`auto|from-font|length/percentage|initial|inherit;

- **`Text-decoration` (line style color thickness)**

  Text-decoration :underline | overline | line-through | none ;

Text Transformation

The `text-transform` property is used to specify uppercase and lowercase letters in a text.

```
text-transform: none|capitalize|uppercase|lowercase|initial|inherit;
```

CSS Text Spacing

CSS Text Indentation, Letter Spacing, Line Height, Word Spacing, and White Space

In this chapter you will learn about the following properties:

- `text-indent` **(where to start the paragraph)**

  `text-indent: *length*|initial|inherit;`

- `letter-spacing`

  `letter-spacing: normal|*length*|initial|inherit;`

- `line-height`

  line-height: normal|*number*|*length*|initial|inherit;

- `Word-spacing`

  word-spacing: normal|*length*|initial|inherit;

- `white-space`

  white-space:
  normal|nowrap|pre|pre-line|pre-wrap|initial|inherit;

  Nowrap = it aligns all the text in a single line.

  Wrap = it aligns the text on the inner side of the box .

  Pre = it prints as the text written in code.

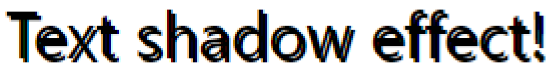
  pre-wrap= it breaks the line in multiple lines when needed.

  Pre-line = it breaks the line when needed and cut space from starting.

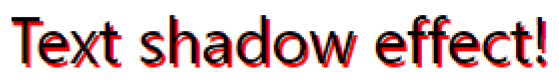
CSS Text Shadow

The `text-shadow` property adds shadow to text.

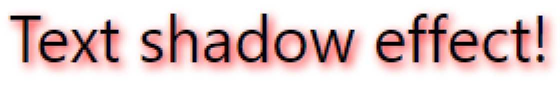
```
text-shadow: 2px 2px;
```



```
text-shadow: 2px 2px red;
```


Text shadow effect!

```
text-shadow: 2px 2px 5px red;
```


Text shadow effect!

CSS Lists

```
list-style-type: value;
```

disc | circle | square | armenian | cjk-ideographic | decimal | decimal-leading-zero | georgian | hebrew | hiragana | hiragana-iroha | katakana | katakana-iroha | lower-alpha | lower-greek | lower-latin | lower-roman | upper-alpha | upper-greek | upper-latin | upper-roman | none | inherit

CSS Tables

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

The `border-collapse` property sets whether table borders should collapse into a single border or be separated as in standard HTML.

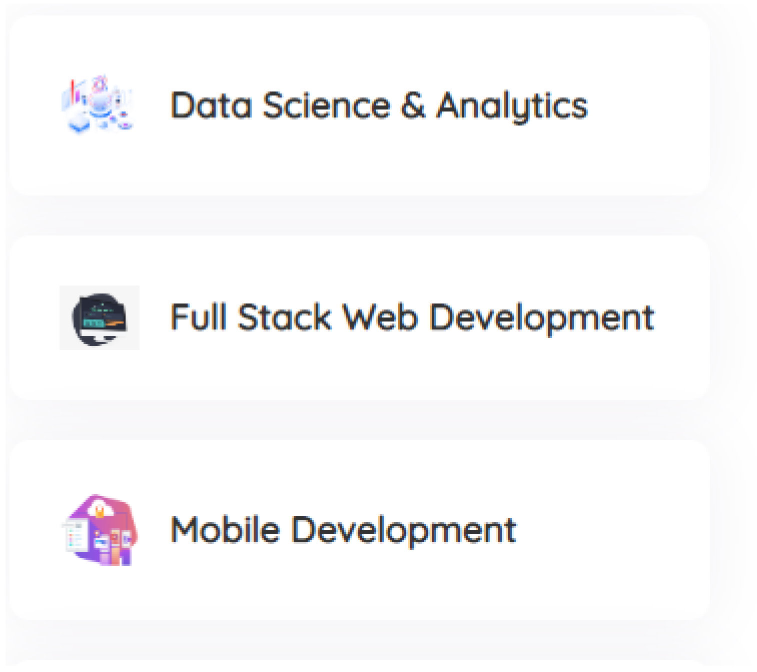
```
border-collapse: separate|collapse|initial|inherit;
```

CSS display Property

```
display: inline | block  | flex | grid | inline-block | inline-flex |
inline-grid | inline-table | list-item | table | table-caption |
table-column-group | table-header-group | table-footer-group |
table-row-group | table-cell | table-column |            table-row |
none | initial  | inherit
```

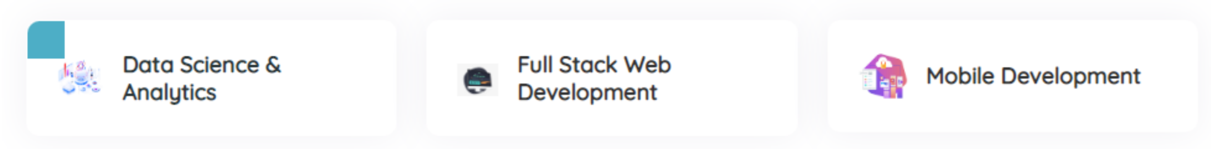
- `Flex-direction : row| column|row-reverse|column-reverse`
- `flex-wrap:nowrap(default)|wrap(it covers its place and  throw contain in the next line)|wrap-reverse`

- Justify-content:center|



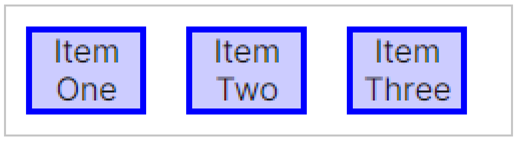
  flex-start|flex-end|space-evenly|space-between|space-around
- Align-item:stretch|flex-end|flex-start|baseline|center
- Align-content:stretch|flex-end|flex-start|baseline|center|space -around|space-between
- Align-self:stretch|flex-end|flex-start|center|baseline
- Flex-grow:numerically but negative is not used and it is work on the box of the inner content

display: flex; /* or inline-flex */

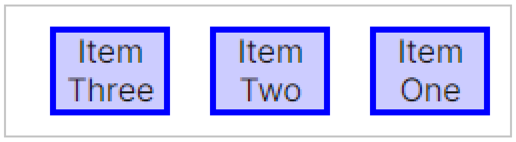


Flex-direction: row | row-reverse | column | column-reverse

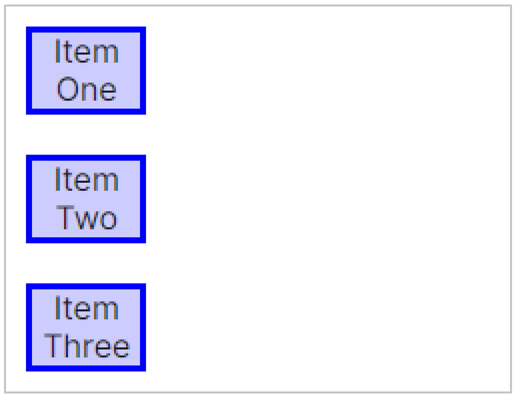
Flex-direction: row;



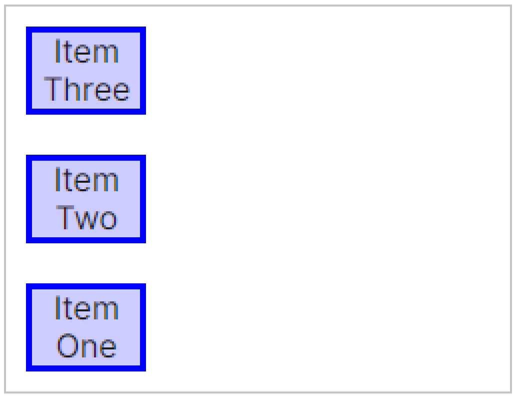
```
Flex-direction:  row-reverse;
```



```
Flex-direction:  column;
```

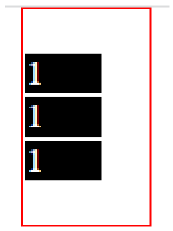


```
Flex-direction:  column-reverse;
```

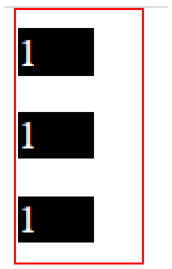


flex-wrap: nowrap | wrap | wrap-reverse;

justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ;

it aligns all the content in the value given  .

align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end ;
(It aligns the items in the center of the box)



align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline ;

Css grid

Display:grid;

grid-templete -columns: auto auto auto ; (you can write the px value also for the width of the columns
Grid–templete-rows: auto auto ; (you can write the px value also for the width of the rows)

CSS Layout - The position Property

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

There are five different position values:

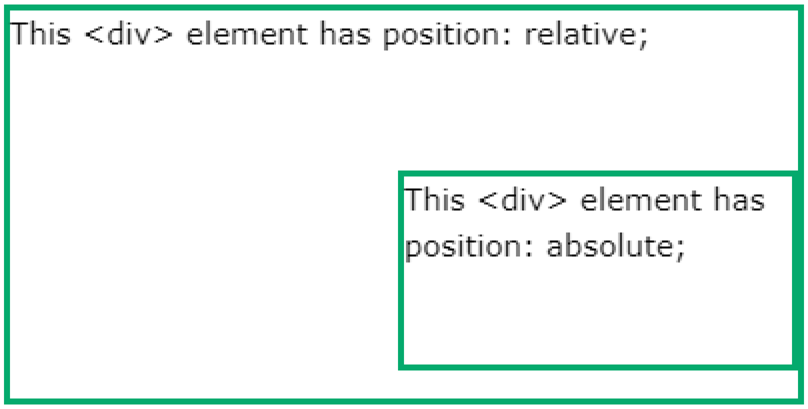
- `Static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

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

This <div> element has position: relative;

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.

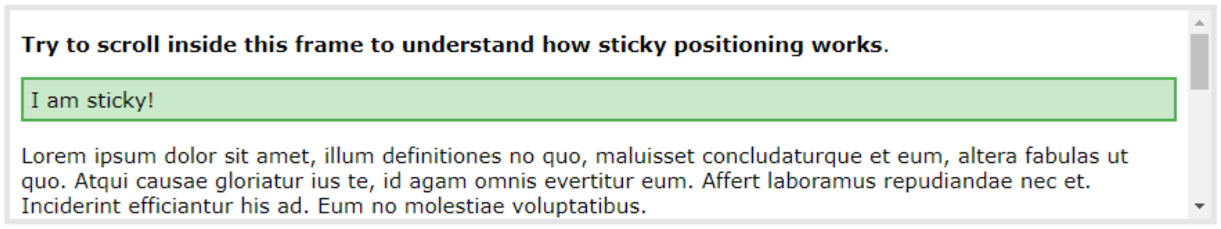
An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).



This <div> element has position: relative;

This <div> element has position: absolute;

An element with `position: sticky;` is positioned based on the user's scroll position.



Try to scroll inside this frame to understand how sticky positioning works.

I am sticky!

Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

CSS Layout - The z-index Property

The `z-index` property specifies the stack order of an element.

Note: **`z-index`** only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of display:flex elements).

```
<img src="image1.jpg" alt="not load" style="position: absolute; top:0;
left:0;z-index:-1; width:100px;">
```

CSS Layout - Overflow

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: visible|hidden|clip|scroll|auto|initial|inherit;

overflow-wrap: normal|anywhere|break-word|initial|inherit;

overflow-x: visible|hidden|scroll|auto|initial|inherit;

overflow-y: visible|hidden|scroll|auto|initial|inherit;
```

CSS Layout - float and clear

The CSS `float` property specifies how an element should float.

The CSS `clear` property specifies what elements can float beside the cleared element and on which side.

The float Property

The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The `float` property can have one of the following values:

- `left` - The element floats to the left of its container
- `right` - The element floats to the right of its container
- `none` - The element does not float (will be displayed just where it occurs in the text). This is default

- `inherit` - The element inherits the float value of its parent

CSS Combinators

A combinator is something that explains the relationship between the selectors.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+) it selects the next first element ;

  (The CSS adjacent sibling selector is used to select the adjacent sibling of an element. It is used to select only those elements which immediately follow the first selector.)

- general sibling selector (~) it selects the next all same element ;

(The CSS general sibling selector is used to select all elements that follow the first element such that both are children of the same parent.)

CSS Pseudo-classes

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

  :hover,focus(is used in inputs only),link(unvisited link),active(it used for the link visit changes),first-child,last-child,only-child,not(),

CSS Pseudo-elements

What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

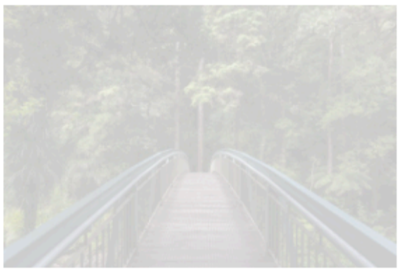
::first-line(it decor the first line.),::first-letter(it decorates the first letter),::before(it adds some content before the element),::after(it adds content after the element),::selection(it decorates the selection process)

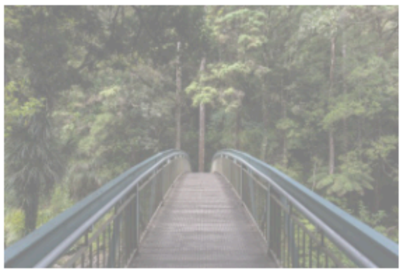
```
a::after/before {

   content: ' (' attr(href) ')';

}

A::after/before{

content:'("hello")';

}
```

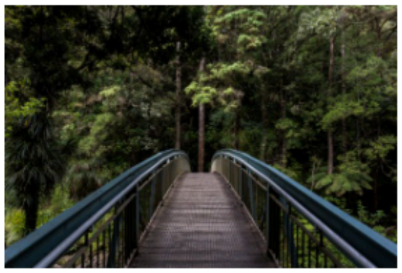
- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

CSS Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element.


opacity 0.2


opacity 0.5

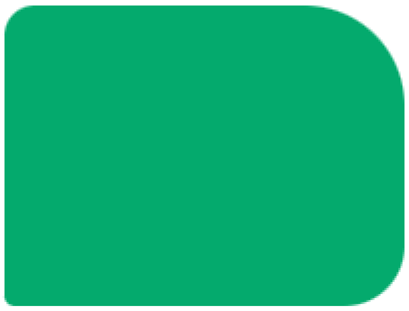

opacity 1
(default)

CSS Advanced

CSS Rounded Corners

The CSS `border-radius` property defines the radius of an element's corners.

border-radius: 15px 50px 30px 5px;



CSS Gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines three types of gradients:

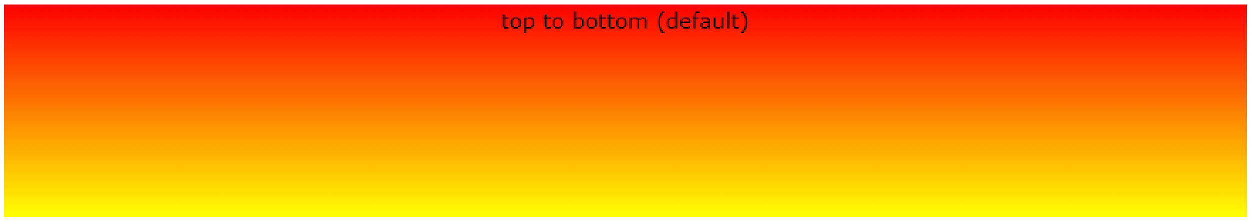
- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)
- Conic Gradients (rotated around a center point)

Syntax

```
background-image: linear-gradient(direction, color-stop1,
color-stop2, ...);
```

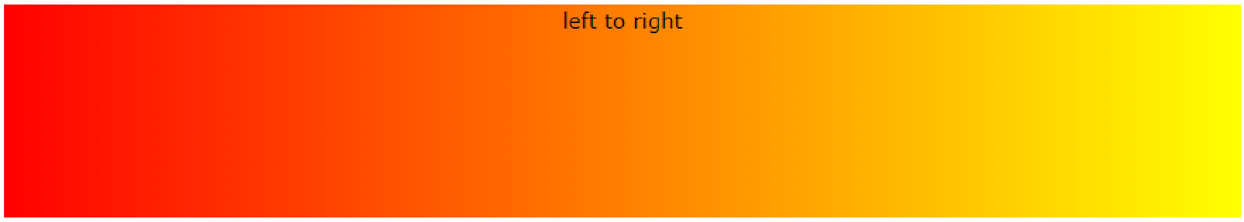
Direction - Top to Bottom (this is default)

```
  background-image: linear-gradient(red, yellow);
```



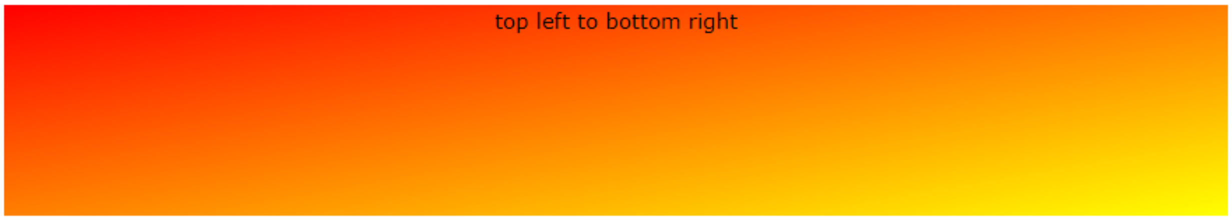
Direction - Left to Right

```
background-image: linear-gradient(to right, red , yellow);
```



Direction - Diagonal

```
background-image: linear-gradient(to bottom right, red, yellow);
```



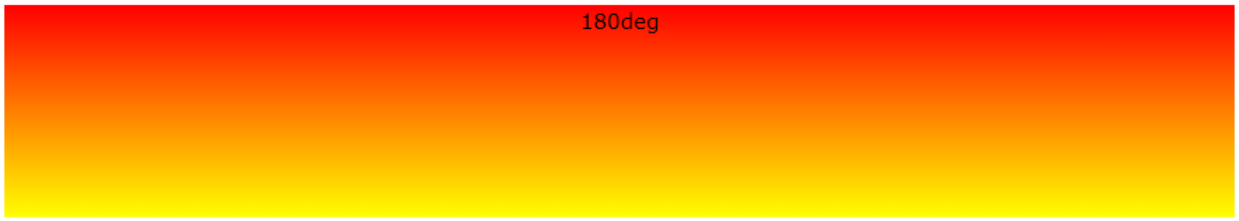
Using Angles

If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.).

Syntax

```
background-image: linear-gradient(angle, color-stop1, color-stop2);
```

```
background-image: linear-gradient(180deg, red, yellow);
```



CSS Radial Gradients

A radial gradient is defined by its center.

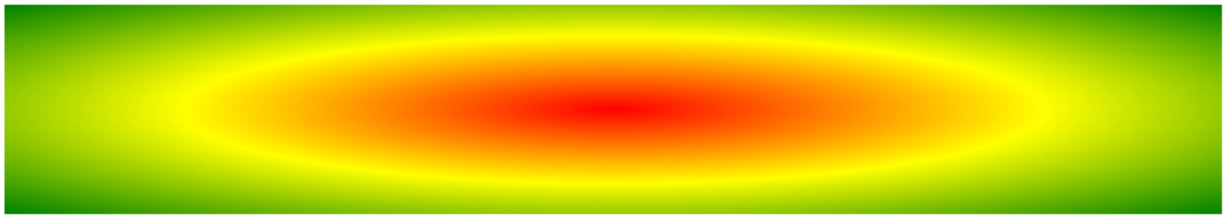
To create a radial gradient you must also define at least two color stops.

Syntax

```
background-image: radial-gradient(shape size at position,
start-color, ..., last-color);
```

Radial Gradient - Evenly Spaced Color Stops (this is default)

```
background-image: radial-gradient(red 5%, yellow 15%, green 60%);
```



Radial Gradient - Differently Spaced Color Stops.

CSS Conic Gradients

A conic gradient is a gradient with color transitions rotated around a center point.

To create a conic gradient you must define at least two colors.

Syntax

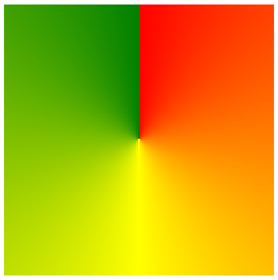
```
background-image: conic-gradient([from angle] [at position,] color
[degree], color [degree], ...);
```

By default, *angle* is 0deg and *position* is center.

If no *degree* is specified, the colors will be spread equally around the center point.

Conic Gradient: Three Colors

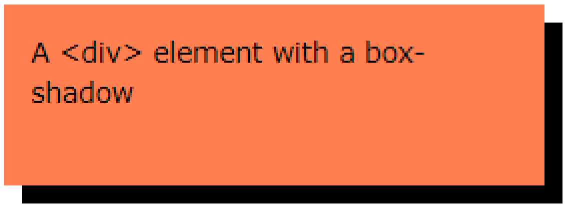
```
background-image: conic-gradient(red, yellow, green);
```



CSS Box Shadow

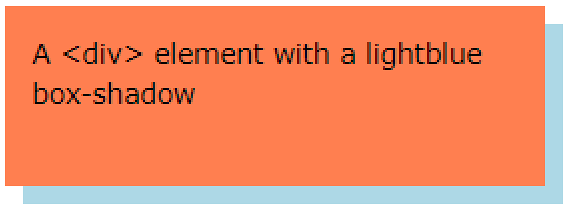
The CSS `box-shadow` property is used to apply one or more shadows to an element.

```
box-shadow: 10px 10px;
```

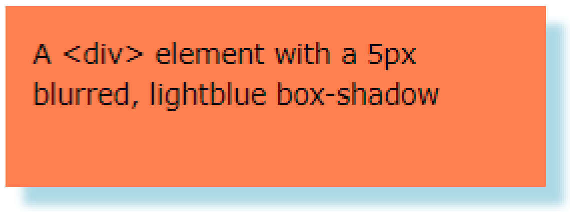


A <div> element with a box-shadow

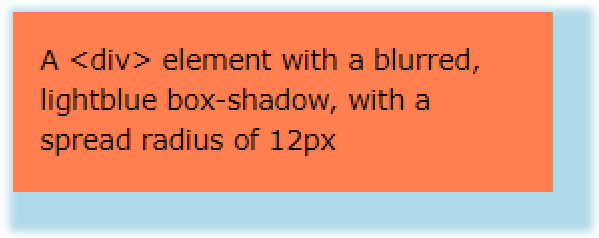
```
box-shadow: 10px 10px lightblue;
```



```
box-shadow: 10px 10px 5px lightblue;
```

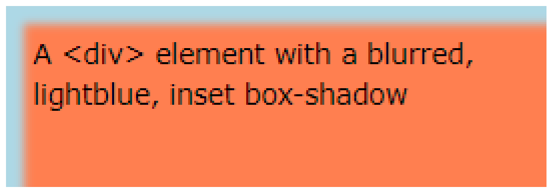


```
box-shadow: 10px 10px 5px 12px lightblue;
```


A <div> element with a blurred, lightblue box-shadow, with a spread radius of 12px

Set the inset Parameter

The `inset` parameter changes the shadow from an outer shadow (outset) to an inner shadow.



CSS Text Effects

CSS Text Overflow, Word Wrap, Line Breaking Rules, and Writing Modes

In this chapter you will learn about the following properties:

- `text-overflow`
- `word-wrap`
- `word-break`
- `writing-mode`

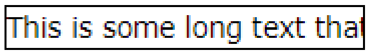
CSS Text Overflow

The CSS `text-overflow` property specifies how overflowed content that is not displayed should be signaled to the user.

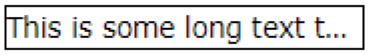
```
text-overflow: clip|ellipsis|initial|inherit;
```

First of all set the white-space:nowrap;for all properties.

It can be clipped:


This is some long text that

or it can be rendered as an ellipsis (...):

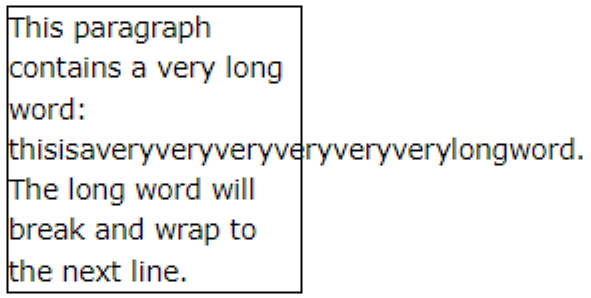

This is some long text t...

CSS Word Wrapping

The CSS `word-wrap` property allows long words to be able to be broken and wrap onto the next line.

```
word-wrap: normal|break-word|initial|inherit;
```

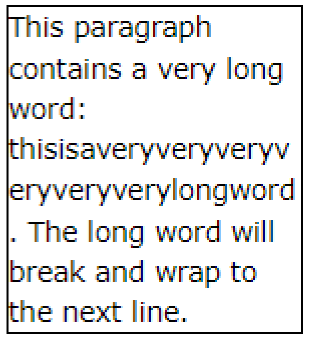
If a word is too long to fit within an area, it expands outside:



This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:

```
word-wrap: break-word;
```

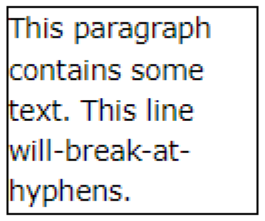


This paragraph contains a very long word: thisisaveryveryveryv eryveryverylongword . The long word will break and wrap to the next line.

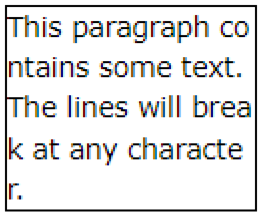
CSS Word Breaking

The CSS `word-break` property specifies line breaking rules.

```
word-break: normal|break-all(it breaks from between also)|keep-all(it
breaks from the words))|break-word(it breaks from the
words)|initial|inherit;
```



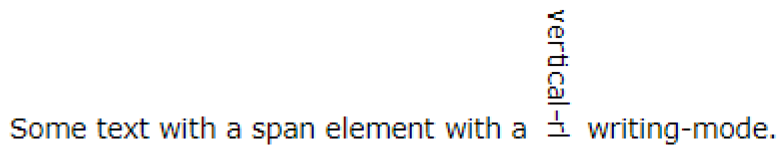
This paragraph contains some text. This line will-break-at- hyphens.


This paragraph co
ntains some text.
The lines will brea
k at any characte
r.

CSS Writing Mode

The CSS `writing-mode` property specifies whether lines of text are laid out horizontally or vertically.

```
writing-mode: horizontal-tb(default)|vertical-rl|vertical-lr;
```


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.

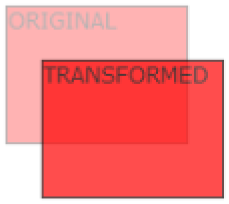
Mouse over the element below to see a 2D transformation:

- `translate()(it transform the box from its actual position)`
- `rotate()(it rotate from it actual position)`
- `scaleX(width)`
- `scaleY(height)`
- `scale(width,height)(it  can change the size of box)`
- `skewX(along x-axis)`
- `skewY(along y-axis)`
- `skew(x and y axis)(10deg ,20deg)`

The translate() Method

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

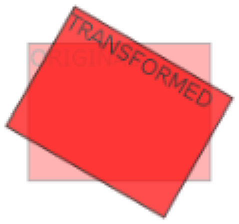
```
transform: translate(50px, 100px);
```



The rotate() Method

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

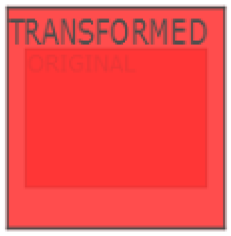
```
transform: rotate(20deg);
```



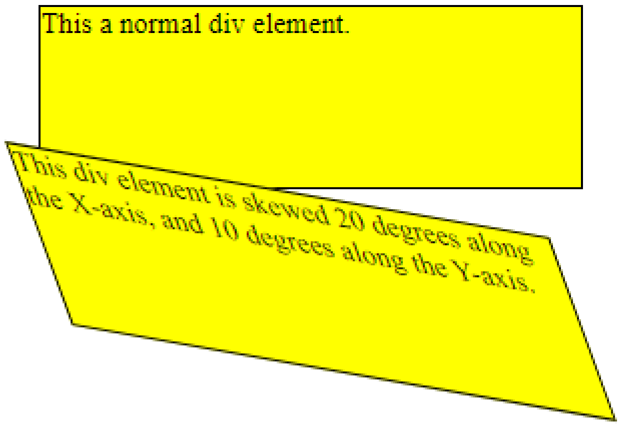
The scale() Method

The `scale()` method increases or decreases the size of an element (according to the parameters given for the width and height).

```
transform: scale(2, 3);
```



The skew() Method


This a normal div element.
This div element is skewed 20 degrees along the X-axis, and 10 degrees along the Y-axis.

CSS Transitions

- `transition(it defines in the main css bar and defines what will be changes )` **`(property duration timing-function delay)`**

- `transition-delay(it is also mentioned in the main section and performs delay )`

- `transition-duration(how much time taken to complete the transition )`

- `transition-property(which property do you want to change)`

- `transition-timing-function(how the transition happened)`

How to Use CSS Transitions?

To create a transition effect, 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.

The following example shows a 100px * 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

Specify the Speed Curve of the Transition

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

The transition-timing-function property can have the following 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
- `cubic-bezier(n,n,n,n)` - lets you define your own values in a cubic-bezier function

CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

- `@keyframes`
- `animation-name`
- `animation-duration`
- `animation-delay`
- `animation-iteration-count`
- `Animation-direction`

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

- `Animation-timing-function  (same as transition)`
- `Animation-fill-mode`

The animation-fill-mode property can have the following values:

- `none` - Default value. Animation will not apply any styles to the element before or after it is executing
- `forwards` - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
- `backwards` - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
- `both` - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions
-

`Shorthaned property`

- `Animation` **`(name  duration  timing-function delay itrationcount direction  fill-mode)`**

What are 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.

The @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.

CSS The object-fit Property

The CSS `object-fit` property is used to specify how an <img> or <video> should be resized to fit its container.

Here is wherethe `object-fit` property comes in. The `object-fit` property can take one of the following values:

- `fill` - This is default. The image is resized to fill the given dimension. If necessary, the image will be stretched or squished to fit
- `contain` - The image keeps its aspect ratio, but is resized to fit within the given dimension
- `cover` - The image keeps its aspect ratio and fills the given dimension. The image will be clipped to fit
- `none` - The image is not resized
- `scale-down` - the image is scaled down to the smallest version of `none` or `contain`

object-fit: cover;

object-fit: contain;

object-fit: fill

object-fit: none;

object-fit: scale-down;

CSS object-position

```
object-position: position|initial|inherit;
```

CSS Variables - The var() Function

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.

Syntax of the var() Function

The `var()` function is used to insert the value of a CSS variable.

The syntax of the `var()` function is as follows:

```
var(--name, value)
```

How var() Works

First of all: CSS variables can have a global or local scope.

Global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.

To create a variable with global scope, declare it inside the `:root` selector. The `:root` selector matches the document's root element.

To create a variable with local scope, declare it inside the selector that is going to use it.

The following example is equal to the example above, but here we use the `var()` function.

First, we declare two global variables (--blue and --white). Then, we use the `var()` function to insert the value of the variables later in the style sheet:

```
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}

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

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

CSS Box Sizing

The `box-sizing` property defines how the width and height of an element are calculated: should they include padding and borders, or not.

```
box-sizing: content-box|border-box|initial|inherit;
```

CSS Media Queries

The **Media query** in CSS is used to create a responsive web design. It means that the view of a web page differs from system to system based on screen or media types. The breakpoint specifies for what device-width size, the content is just starting to break or deform.

Media queries can be used to check many things:

- width and height of the viewport
- width and height of the device
- Orientation
- Resolution

A media query consist of a media type that can contain one or more expression which can be either true or false. The result of the query is true if the specified media matches the type of device the document is displayed on. If the media query is true then a style sheet is applied.

```
@media not|only mediatype and (expressions) {
  CSS-Code;
}
```

Responsive Web Design - The Viewport

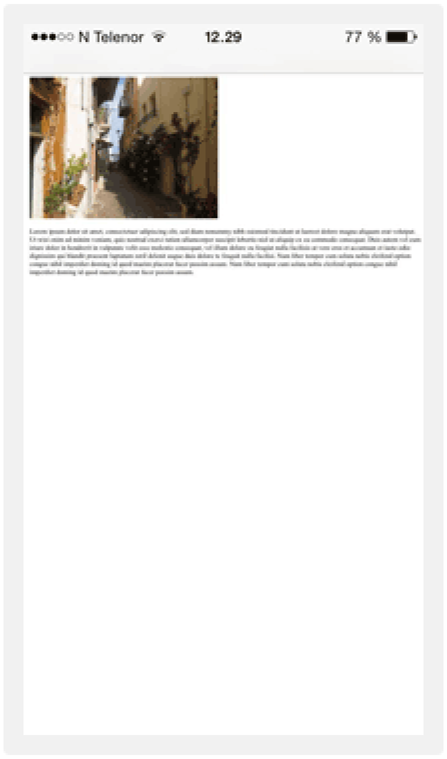
What is The Viewport?

The viewport is the user's visible area of a web page.

The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

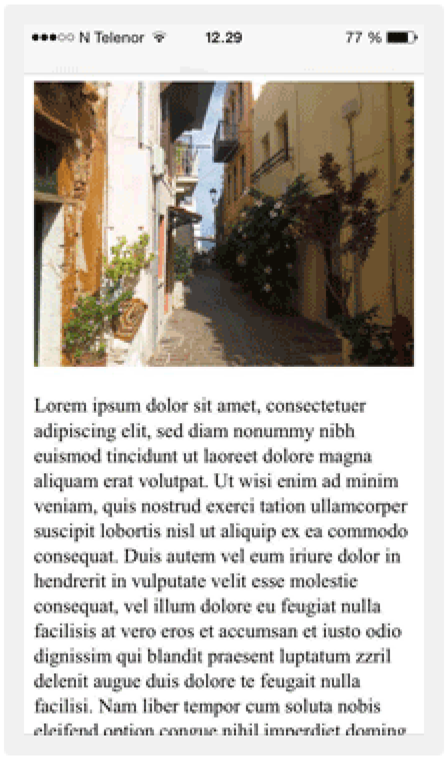
Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

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



Without the viewport meta tag

```
@media only screen and (max-width: value) {

Selector{

  Property: Value;

  }

}
```

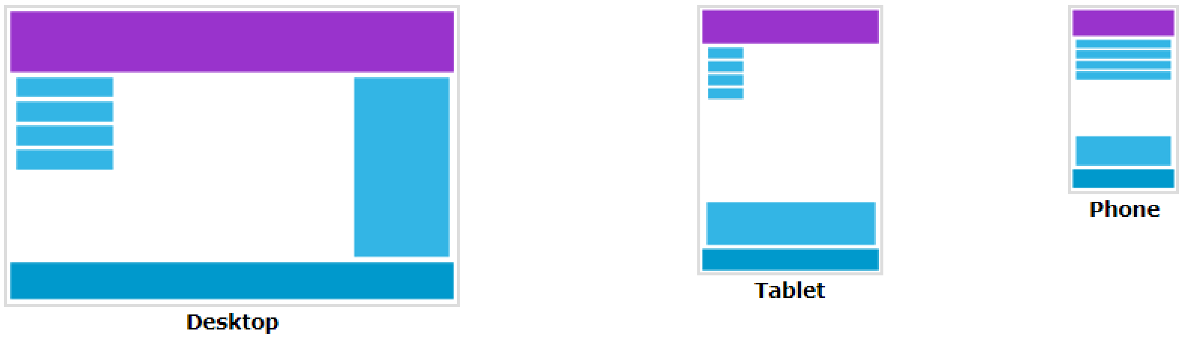

●●●○○ N Telenor 12.29 77 %

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

With the viewport meta tag

Add a Breakpoint

Earlier in this tutorial we made a web page with rows and columns, and it was responsive, but it did not look good on a small screen.



```
@media only screen and (max-width: 1199px) {

      Selectors{

            Property:value;

      }

}

@media only screen and (max-width: 1024px) {

      Selectors{

            Property:value;

      }

}

@media only screen and (max-width: 991px) {

      Selectors{

            Property:value;

      }

}

@media only screen and (max-width: 767px) {

      Selectors{

            Property:value;

      }

}
```

CSS units – %, em, rem, px, vh, vw

- **% –** The % unit is used to set the font-size relative to the current font-size.
- **em –** It is used to set the relative size. It is relative to the font-size of the element.

  Note: Here 2em means 2times the size of the current font.
- **rem –** Relative to the browser base font-size.
- **px –** It defines the font-size in terms of pixels. (96px = 1in)
- **vh –** Relative to 1% of the height of the viewport.
- **vw –** Relative to 1% of the width of the viewport.