CSS Introduction

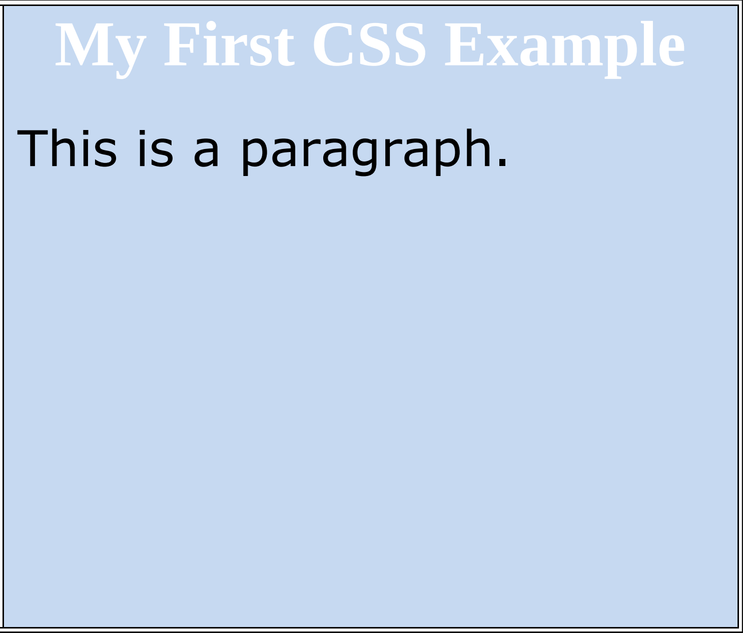
What is CSS?

- 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.

```
<head>
<style>
body {
  background-color: lightblue;
}
h1 {
  color: white;
  text-align: center;
}
p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
</body>
```



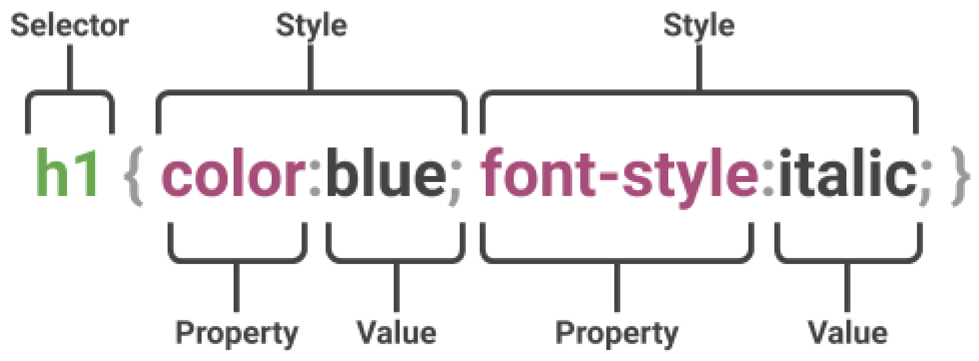
My First CSS Example

This is a paragraph.

1).CSS Syntax,Selectors,How To Add CSS Comments

A).Syntax

The selector points to the HTML element you want to style.
The declaration block contains one or more declarations separated by semicolons.
Each declaration includes a CSS property name and a value, separated by a colon.
Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.



B). Selector

p {} -<p> elements on the page
#para1 {} -element with id="para1":
.center {} -elements with class="center" will
p.center {} -elements with class="center"
<p class="center large">- element will be styled according to class="center" and to class="large":
* {} -universal selector (*)
h1 {}
h2 {} } h1, h2, p {} - Grouping
p {}

C). How To Add CSS Comments

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.
Comments are ignored by browsers.
A CSS comment is placed inside the <style> element, and starts with /* and ends with */:

2).CSS Colors,Backgrounds,Borders

A).CSS Color Names

B)CSS Backgrounds

The CSS background properties are used to add background effects for elements.

a).CSS background-color

The background-color property specifies the background color of an element.

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

With CSS, a color is most often specified by:

- a valid color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Look at CSS Color Values for a complete list of possible color values.

b)Opacity / Transparency

The opacity property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent:

```
div {
  background-color: green;
  opacity: 0.3;
}
```

c).Transparency using RGBA

```
div {
  background: rgba(0, 128, 0, 0.3) /* Green
background with 30% opacity */
}
```

d).Background Image

The background-image property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

```
body {
  background-image: url("paper.gif");
}
```

e).background-repeat

```
body {
  background-image: url("gradient_bg.png");
}
```

If the image above is repeated only horizontally (background-repeat: repeat-x;), the background will look better:

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

Showing the background image only once is also specified by the background-repeat property:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
```

The background-position property is used to specify the position of the background image.

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

f). 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("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

Specify that the background image should scroll with the rest of the page:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: scroll;
}
```

Shorthand property

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

C).Borders

a).Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

b). Border Width

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

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick:

```
p.one {
  border-style: solid;
  border-width: 5px;
}

p.two {
  border-style: solid;
  border-width: medium;
}

p.three {
  border-style: dotted;
  border-width: 2px;
}
```

Specific Side Widths

The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border):

```
p.one {
  border-style: solid;
  border-width: 5px 20px; /* 5px top and bottom, 20px
on the sides */
}

p.two {
  border-style: solid;
  border-width: 20px 5px; /* 20px top and bottom, 5px
on the sides */
}

p.three {
  border-style: solid;
  border-width: 25px 10px 4px 35px; /* 25px top, 10px
right, 4px bottom and 35px left */
}
```

Rounded Borders

The border-radius property is used to add rounded borders to an element:

```
border-radius: 5px;
```

3).CSS Margins, Padding, Height/Width

A).Margins

The CSS margin properties are used to create space around elements, outside of any defined borders.
With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

a).Individual Sides

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

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

All the margin properties can have the following values:

- auto - the browser calculates the margin
- length - specifies a margin in px, pt, cm, etc.
- % - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

Tip: Negative values are allowed.

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

Shorthand Property

To shorten the code, it is possible to specify all the margin properties in one property.
So, here is how it works:
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

```
p {
  margin: 25px 50px 75px 100px;
}
```

margin: 25px 50px 75px;

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

```
p {
  margin: 25px 50px 75px;
}
```

If the margin property has two values:

margin: 25px 50px;

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

```
p {
  margin: 25px 50px;
}
```

If the margin property has one value:

margin: 25px;

  - all four margins are 25px

```
p {
  margin: 25px;
}
```

The auto Value

You can set the margin property to auto to horizontally center the element within its container.
The element will then take up the specified width, and the remaining space will be split equally between the left and right margins.

```
div {
  width: 300px;
  margin: auto;
  border: 1px solid red;
}
```

The inherit Value

This example lets the left margin of the <p class="ex1"> element be inherited from the parent element (<div>):

```
div {
  border: 1px solid red;
  margin-left: 100px;
}

p.ex1 {
  margin-left: inherit;
}
```

B).Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Individual Sides

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

- padding-top
- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element
- inherit - *specifies that the padding should be inherited from the parent element*

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

Shorthand Property

To shorten the code, it is possible to specify all the padding properties in one 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

```
div {
  padding: 25px 50px 75px 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

```
div {
  padding: 25px 50px 75px;
}
```

If the padding property has two values:

padding: 25px 50px;

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

```
div {
  padding: 25px 50px;
}
```

If the padding property has one value:

padding: 25px;

- all four paddings are 25px

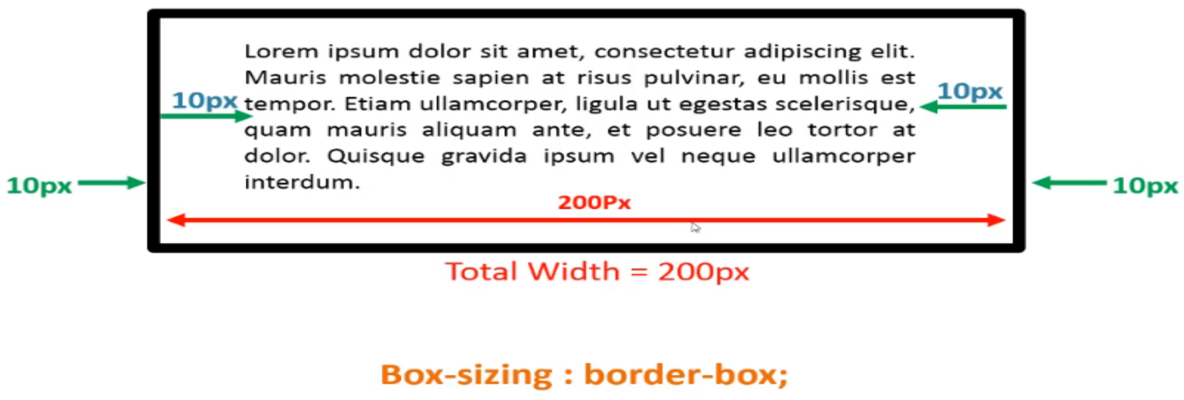
```
div {
  padding: 25px;
}
```

box-sizing-border-box

To keep the width at 300px, no matter the amount of padding, you can use the box-sizing property. This causes the element to maintain its actual width; if you increase the padding, the available content space will decrease.

Use the box-sizing property to keep the width at 300px, no matter the amount of padding:

```
div {
  width: 300px;
  padding: 25px;
  box-sizing: border-box;
}
```


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris molestie sapien at risus pulvinar, eu mollis est tempor. Etiam ullamcorper, ligula ut egestas scelerisque, quam mauris aliquam ante, et posuere leo tortor at dolor. Quisque gravida ipsum vel neque ullamcorper interdum.

C).Height, 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.

div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}	

max-width

Note: If you for some reason use both the width property and the max-width property on the same element, and the value of the width property is larger than the max-width property; the max-width property will be used (and the width property will be ignored).

div {
  max-width: 600px;
  height: 100px;
  background-color: powderblue;
}	

4).CSS, Outline, Text,Fonts

A).Outline

CSS has the following outline properties:

- outline-style
- outline-color
- outline-width
- outline-offset
- outline

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

p.dotted {outline-style: dotted;}	

Outline Width

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

- thin (typically 1px)
- medium (typically 3px)
- thick (typically 5px)
- A specific size (in px, pt, cm, em, etc)

Outline Color

The outline-color property is used to set the color of the outline.

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

Shorthand property

p.ex3 {outline: 5px solid yellow;}	

Outline Offset

outline-offset: 15px;	

B).Text

Text Color

```
body {
  background-color: lightgrey;
  color: blue;
}
```

a).Alignment

In this chapter you will learn about the following properties:

- text-align
- text-align-last
- direction
- unicode-bidi
- vertical-align

Text Alignment

The text-align property is used to set the horizontal alignment of a text.
A text can be left or right aligned, centered, or justified.
The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

```
h1 {
  text-align: center;
}

h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
```

When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

```
div {
  text-align: justify;
}
```

Text Align Last

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

```
p.a {
  text-align-last: right;
}
p.b {
  text-align-last: center;
}
p.c {
  text-align-last: 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.a {
  vertical-align: baseline;
}
img.b {
  vertical-align: text-top;
}
img.c {
  vertical-align: text-bottom;
}
img.d {
  vertical-align: sub;
}
img.e {
  vertical-align: super;
}
```

b).Text Decoration

In this chapter you will learn about the following properties:

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

The text-decoration-line property is used to add a decoration line to text.

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

h2 {
  text-decoration-line: line-through;
}

h3 {
  text-decoration-line: underline;
}

p {
  text-decoration-line: 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;
}

h2 {
  text-decoration-line: line-through;
  text-decoration-color: blue;
}
```

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;
}
p.ex1 {
  text-decoration-line: underline;
  text-decoration-style: dashed;
}

p.ex2 {
  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;
}
h3 {
  text-decoration-line: underline;
  text-decoration-thickness: 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)

```
h1 {
  text-decoration: underline;
}

h2 {
  text-decoration: underline red;
}

h3 {
  text-decoration: underline red double;
}

p {
  text-decoration: underline red double 5px;
}
```

A Small Tip

All links in HTML are underlined by default. Sometimes you see that links are styled with no underline. The text-decoration: none; is used to remove the underline from links, like this:

```
a {
  text-decoration: none;
}
```

c).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:

```
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
```

Using the text-transform property

THIS TEXT IS TRANSFORMED TO UPPERCASE.

this text is transformed to lowercase.

This Text Is Capitalized.

d). Spacing

In this chapter you will learn about the following properties:

- text-indent
- letter-spacing
- line-height
- word-spacing
- white-space

Indentation

The text-indent property is used to specify the indentation of the first line of a text:

p {
  text-indent: 50px;
}
	Using text-indent
	
	In my younger and more vulnerable years my father gave me s

Letter Spacing

The letter-spacing property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

h1 {
  letter-spacing: 5px;
}
 	
h2 {
  letter-spacing: -2px;
}
	Using letter-spacing
	
	More Money is more happiness
	
	No money get happiness

Line Height

The line-height property is used to specify the space between lines:

p.small {
  line-height: 0.8;
}
 	
p.big {
  line-height: 1.8;
}	

Word Spacing

The word-spacing property is used to specify the space between the words in a text.
The following example demonstrates how to increase or decrease the space between words:

p.one {
  word-spacing: 10px;
}
 	
p.two {
  word-spacing: -2px;
}	

White Space

The white-space property specifies how white-space inside an element is handled.

This example demonstrates how to disable text wrapping inside an element:

p {
  white-space: nowrap;
}	

e).Text Shadow

The text-shadow property adds shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

h1 {
  text-shadow: 2px 2px;
}	

Next, add a color (red) to the shadow:

h1 {
  text-shadow: 2px 2px red;
}	

Then, add a blur effect (5px) to the shadow:

h1 {
  text-shadow: 2px 2px 5px red;
}	

Example

h1 {
  color: white;
  text-shadow: 2px 2px 4px #000000;
}
h1 {
  text-shadow: 0 0 3px #ff0000;
}
h1 {
  text-shadow: 0 0 3px #ff0000, 0 0 5px #0000ff;
}
h1 {
  color: white;
  text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;
}	

C).Fonts

Choosing the right font for your website is important!

a).Font Selection is Important

Choosing the right font has a huge impact on how the readers experience a website.

The right font can create a strong identity for your brand.

Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

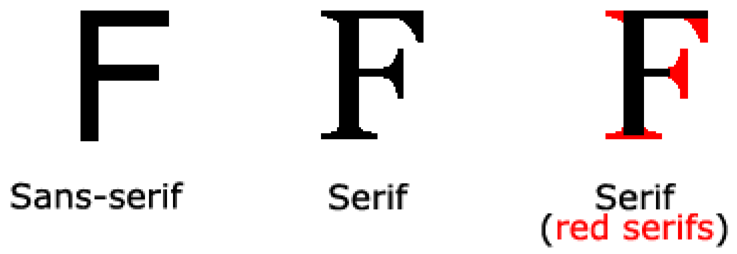
Generic Font Families

In CSS there are five generic font families:

1. Serif fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. Sans-serif fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. Monospace fonts - here all the letters have the same fixed width. They create a mechanical look.
4. Cursive fonts imitate human handwriting.
5. Fantasy fonts are decorative/playful fonts.

All the different font names belong to one of the generic font families.

Difference Between Serif and Sans-serif Fonts



The CSS font-family Property

In CSS, we use the font-family property to specify the font of a text.

```
p1 {
  font-family: "Times New Roman", Times, serif;
}
```

b).Web Safe Fonts?

Web safe fonts are fonts that are universally installed across all browsers and devices.

Fallback Fonts

However, 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.

This means that you should add a list of similar "backup fonts" in the font-family property. If the first font does not work, the browser will try the next one, and the next one, and so on. Always end the list with a generic font family name.

```
p {
font-family: Tahoma, Verdana, sans-serif;
}
```

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)

➢ Times New Roman (serif)

Times New Roman is one of the most recognizable fonts in the world. It looks professional and is used in many newspapers and "news" websites. It is also the primary font for Windows devices and applications.

Lorem ipsum dolor sit amet

➢ Georgia (serif)

Georgia is an elegant serif font. It is very readable at different font sizes, so it is a good candidate for mobile-responsive design.

Lorem ipsum dolor sit amet

➢ Garamond (serif)

Garamond is a classical font used for many printed books. It has a timeless look and good readability.

Lorem ipsum dolor sit amet

- Courier New (monospace)

Courier New is the most widely used monospace serif font. Courier New is often used with coding displays, and many email providers use it as their default font. Courier New is also the standard font for movie screenplays.

Lorem ipsum dolor sit amet

- Brush Script MT (cursive)

The Brush Script MT font was designed to mimic handwriting. It is elegant and sophisticated, but can be hard to read. Use it carefully.

Lorem ipsum dolor sit amet

d).CSS Font Style

The font-style property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

p.normal {
  font-style: normal;
}
 	
p.italic {
  font-style: italic;
}
 	
p.oblique {
  font-style: oblique;
}	

Font Weight

The font-weight property specifies the weight of a font:

p.normal {
  font-weight: normal;
}
 	
p.thick {
  font-weight: bold;
}	

Font Variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

```
p.normal {
  font-variant: normal;
}

p.small {
  font-variant: small-caps;
}
```

e).Font Size

The font-size property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.

The font-size value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

16px (16px=1em).

Set Font Size With Pixels

```
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
```

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: pixels/16=em

```
h1 {
  font-size: 2.5em; /* 40px/16=2.5em */
}

h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
}

p {
  font-size: 0.875em; /* 14px/16=0.875em */
}
```

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

```
body {
  font-size: 100%;
}

h1 {
  font-size: 2.5em;
}

h2 {
  font-size: 1.875em;
}

p {
  font-size: 0.875em;
}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

Responsive Font Size

The text size can be set with a vw unit, which means the "viewport width".

That way the text size will follow the size of the browser window:

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

f).Great Font Pairings

Great font pairings are essential to great design.

Here are some basic rules to create great font pairings:

1. Complement

It is always safe to find font pairings that complement one another.

A great font combination should harmonize, without being too similar or too different.

2. Use Font Superfamilies

A font superfamily is a set of fonts designed to work well together. So, using different fonts within the same superfamily is safe.

For example, the Lucida superfamily contains the following fonts: Lucida Sans, Lucida Serif, Lucida Typewriter Sans, Lucida Typewriter Serif and Lucida Math.

3. Contrast is King

Two fonts that are too similar will often conflict. However, contrasts, done the right way, brings out the best in each font.

Example: Combining serif with sans serif is a well known combination.

A strong superfamily includes both serif and sans serif variations of the same font (e.g. Lucida and Lucida Sans).

4. Choose Only One Boss

One font should be the boss. This establishes a hierarchy for the fonts on your page. This can be achieved by varying the size, weight and color.

```
body {
  background-color: black;
  font-family: Verdana, sans-serif;
  font-size: 16px;
  color: gray;
}

h1 {
  font-family: Georgia, serif;
  font-size: 60px;
  color: white;
}
```

Georgia and Verdana

Georgia and Verdana is a classic combination. It also sticks to the web safe font standards:

Use the "Georgia" font for headings, and "Verdana" for text:

Helvetica and Garamond

Helvetica and Garamond is another classic combination that uses web safe fonts:

Use the "Helvetica" font for headings, and "Garamond" for text:

Popular Google Font Pairings

If you do not want to use standard fonts in HTML, you can use Google Fonts.

Google Fonts are free to use, and have more than 1000 fonts to choose from.

Below are some popular Google Web Font Pairings.

Merriweather and Open Sans

Use the "Merriweather" font for headings, and "Open Sans" for text:

Ubuntu and Lora

Use the "Ubuntu" font for headings, and "Lora" for text:

Abril Fatface and Poppins

Use the "Abril Fatface" font for headings, and "Poppins" for text:

Cinzel and Fauna One

Use the "Cinzel" font for headings, and "Fauna One" for text:

Fjalla One and Libre Baskerville

Use the "Fjalla One" font for headings, and "Libre Baskerville" for text:

Space Mono and Muli

Use the "Space Mono" font for headings, and "Muli" for text:

Spectral and Rubik

Use the "Spectral" font for headings, and "Rubik" for text:

Oswald and Noto Sans

Use the "Oswald" font for headings, and "Noto Sans" for text:

g).Font Property

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

p.a {
  font: 20px Arial, sans-serif;
}
 	
p.b {
  font: italic small-caps bold 12px/30px Georgia, serif;
}	

5).CSS, Icons, Links, Lists, Tables

A).Icons

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

The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

Add the name of the specified icon class to any inline HTML element (like <i> or <span>).

All the icons in the icon libraries below, are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

Font Awesome Icons

To use the Font Awesome icons, go to fontawesome.com, sign in, and get a code to add in the <head> section of your HTML page:

```
<script src="https://kit.fontawesome.com/yourcode.js" crossorigin ="anonymous"></script>
```

Read more about how to get started with Font Awesome in our Font Awesome 5 tutorial.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous">
</script>
</head>
<body>
<i class="fas fa-cloud"></i>
<i class="fas fa-heart"></i>
<i class="fas fa-car"></i>
<i class="fas fa-file"></i>
<i class="fas fa-bars"></i>

</body>
</html>
```

Some Font Awesome icons

☁ ♥ 🚗 📄 ☰

Google Icons

To use the Google icons, add the following line inside the <head> section of your HTML page:

```
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>

<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>

</body>
</html>
```

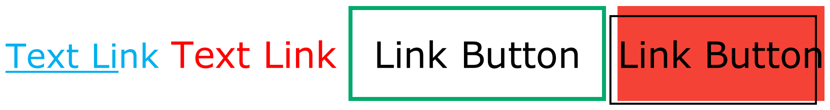
Google icon library

Some Google icons:

☁ ♥ 📎 💻 🚦

B).Links

With CSS, links can be styled in many different ways.


Text Link Text Link Link Button Link Button

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

```
a {
  color: hotpink;
}
```

In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

```
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

```
a:link {
  text-decoration: none;
}

a:visited {
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

a:active {
  text-decoration: underline;
}
```

Background Color

The background-color property can be used to specify a background color for links:

```
a:link {
  background-color: yellow;
}

a:visited {
  background-color: cyan;
}

a:hover {
  background-color: lightgreen;
}

a:active {
  background-color: hotpink;
}
```

Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

```
a:link, a:visited {
  background-color: #f44336;
  color: white;
  padding: 14px 25px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
}

a:hover, a:active {
  background-color: red;
}
```

```
a:link, a:visited {
  background-color: white;
  color: black;
  border: 2px solid green;
  padding: 10px 20px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
}

a:hover, a:active {
  background-color: green;
  color: white;
}
```

This example demonstrates the different types of cursors (can be useful for links):

```
<span style="cursor: auto">auto</span><br>
<span style="cursor: crosshair">crosshair</span><br>
<span style="cursor: default">default</span><br>
<span style="cursor: e-resize">e-resize</span><br>
<span style="cursor: help">help</span><br>
<span style="cursor: move">move</span><br>
<span style="cursor: n-resize">n-resize</span><br>
<span style="cursor: ne-resize">ne-resize</span><br>
<span style="cursor: nw-resize">nw-resize</span><br>
<span style="cursor: pointer">pointer</span><br>
<span style="cursor: progress">progress</span><br>
<span style="cursor: s-resize">s-resize</span><br>
<span style="cursor: se-resize">se-resize</span><br>
<span style="cursor: sw-resize">sw-resize</span><br>
<span style="cursor: text">text</span><br>
<span style="cursor: w-resize">w-resize</span><br>
<span style="cursor: wait">wait</span>
```

B).Lists

The list-style-type property specifies the type of list item marker.
The list-style-image property specifies an image as the list item marker:

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:

```
ul.a {
  list-style-position: outside;
}

ul.b {
  list-style-position: inside;
}
```

Remove Default Settings

The 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;
}
```

B).Table

Striped Tables

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

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

Responsive Table

```
<div style="overflow-x:auto;">
```

6).CSS Display, position, z-index, Overflow, Float and Clear

A).display Property

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.

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).
The <div> element is a block-level element.
Examples of block-level elements:

- <div>
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

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 <span> element inside a paragraph.
Examples of inline elements:

- <span>
- <a>
- <img>

Display: none;

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.
The <script> element uses display: none; as default.

Override The Default Display Value

As mentioned, every element has a default display value. However, you can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline <li> elements for horizontal menus:

```
li {
  display: inline;
}
```

The following exa mple displays <span> elements as block elements:

```
span {
  display: block;
}
```

The following example displays <a> elements as block elements:

```
a {
  display: block;
}
```

Hide an Element - display:none or visibility:hidden?

Hiding an element can be done by setting the display property to none. The element will be hidden, and the page will be displayed as if the element is not there:

```
h1.hidden {
  display: none;
}
```

visibility:hidden; also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

```
h1.hidden {
  visibility: hidden;
}
```

B). The position Property

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

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

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default.

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

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

This <div> element has position: static;

```
div.static {
  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.

This <div> element has position: relative;

Here is the CSS that is used:

```
div.relative {
  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.

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

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

```
div.fixed {
  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 (instead of positioned relative to the viewport, like fixed).

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.

Here is a simple example:

This <div> element has position: relative;

This <div> element has position: absolute;

Here is the CSS that is used:

```
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).

Note: Internet Explorer does not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of top, right, bottom or left for sticky positioning to work.

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

```
div.sticky {
  position: -webkit-sticky; /* Safari */
  position: sticky;
  top: 0;
  background-color: green;
  border: 2px solid #4CAF50;
}
```

C).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:

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

Another z-index Example

```
<html>
<head>
<style>
.container {
  position: relative;
}

.black-box {
  position: relative;
  z-index: 1;
  border: 2px solid black;
  height: 100px;
  margin: 30px;
}

.gray-box {
  position: absolute;
  z-index: 3;
  background: lightgray;
  height: 60px;
  width: 70%;
  left: 50px;
  top: 50px;
}

.green-box {
  position: absolute;
  z-index: 2;
  background: lightgreen;
  width: 35%;
  left: 270px;
  top: -15px;
  height: 100px;
}
</style>
</head>
<body>

<div class="container">
  <div class="black-box">Black box</div>
  <div class="gray-box">Gray box</div>
  <div class="green-box">Green box</div>
</div>

</body>
</html>
```

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: visible

By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:

```
div {
  width: 200px;
  height: 65px;
  background-color: coral;
  overflow: visible;
}
```

overflow: hidden

With the hidden value, the overflow is clipped, and the rest of the content is hidden:

```
div {
  overflow: hidden;
}
```

overflow: scroll

Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

```
div {
  overflow: scroll;
}
```

overflow: auto

The auto value is similar to scroll, but it adds scrollbars only when necessary:

```
div {
  overflow: auto;
}
```

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.

```
div {
  overflow-x: hidden; /* Hide horizontal scrollbar */
  overflow-y: scroll; /* Add vertical scrollbar */
}
```

D). float and clear

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

In its simplest use, the float property can be used to wrap text around images.

```
img {
  float: right;
}
img {
  float: left;
}
img {
  float: none;
}
```

7).CSS display: inline-block, Align Elements, Combinators, Pseudo-classes, Pseudo-elements

A). display: inline-block

Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.

Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.

Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

The following example shows the different behavior of display: inline, display: inline-block and display: block:

Example

Navigation

One common use for display: inline-block is to display list items horizontally instead of vertically. The following example creates horizontal navigation links:

Example

B).Align Elements

Center Align Elements

To horizontally center a block element (like <div>), use margin: auto;

Setting the width of the element will prevent it from stretching out to the edges of its container.

The element will then take up the specified width, and the remaining space will be split equally between the two margins:

This div element is centered.

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  margin: auto;
  width: 60%;
  border: 3px solid #73AD21;
  padding: 10px;
}
</style>
</head>
<body>
```

```
<h2>Center Align Elements</h2>
<p>To horizontally center a block element (like div),
use margin: auto;</p>

<div class="center">
  <p>Hello World!</p>
</div>

</body>
</html>
```

Center Align Text

To just center the text inside an element, use text-align: center;

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  border: 3px solid green;
}
</style>
</head>
<body>

<h2>Center Text</h2>

<div class="center">
  <p>This text is centered.</p>
</div>

</body>
</html>
```

Center an Image

To center an image, set left and right margin to auto and make it into a block element:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  display: block;
  margin-left: auto;
  margin-right: auto;
}
</style>
</head>
<body>

<h2>Center an Image</h2>
<p>To center an image, set left and right margin to
auto, and make it into a block element.</p>

<img src="paris.jpg" alt="Paris" style="width:40%">
```

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

C).Combinators

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all <p> elem

```
div p {
  background-color: yellow;
}
```

Child Selector (>)

The child selector selects all elements that are the children of a specified element.

The following example selects all <p> elements that are children of a <div> element:

```
div > p {
  background-color: yellow;
}
```

Adjacent Sibling Selector (+)

The adjacent sibling selector is used to select an element that is directly after another specific element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects the first <p> element that are placed immediately after <div> elements:

```
div + p {
  background-color: yellow;
}
```

General Sibling Selector (~)

The general sibling selector selects all elements that are next siblings of a specified element.

The following example selects all <p> elements that are next siblings of <div> elements:

```
div ~ p {
  background-color: yellow;
}
```

D).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

Anchor Pseudo-classes

```
* unvisited link */
a:link {
  color: #FF0000;
}

/* visited link */
a:visited {
  color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}

/* selected link */
a:active {
  color: #0000FF;
}
```

Pseudo-classes and HTML Classes

Pseudo-classes can be combined with HTML classes:

When you hover over the link in the example, it will change color:

```
a.highlight:hover {
  color: #ff0000;
}
```

Hover

CSS - The :first-child Pseudo-class

The :first-child pseudo-class matches a specified element that is the first child of another element.

Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

```
p:first-child {
  color: blue;
}
```

Match the first <i> element in all <p> elements

In the following example, the selector matches the first <i> element in all <p> elements:

```
p i:first-child {
  color: blue;
}
```

Match all <i> elements in all first child <p> elements

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

```
p:first-child i {
  color: blue;
}
```

More

E).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:

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

The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all <p> elements:

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

The following properties apply to the ::first-line pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

The ::first-letter Pseudo-element

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

The following example formats the first letter of the text in all <p> elements:

```
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
```

The following properties apply to the ::first-letter pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and HTML Classes

Pseudo-elements can be combined with HTML classes:

```
p.intro::first-letter {
  color: #ff0000;
  font-size: 200%;
}
```

CSS - The ::before Pseudo-element

The ::before pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each <h1> element:

```
h1::before {
  content: url(smiley.gif);
}
```

CSS - The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after the content of each <h1> element:

```
h1::after {
  content: url(smiley.gif);
}
```

CSS - The ::marker Pseudo-element

The ::marker pseudo-element selects the markers of list items.

The following example styles the markers of list items:

```
::marker {
  color: red;
  font-size: 23px;
}
```

CSS - The ::selection Pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to ::selection: color, background, cursor, and outline.

The following example makes the selected text red on a yellow background:

```
::selection {
  color: red;
  background: yellow;
}
```

8).CSS Opacity, Navigation Bar, Dropdowns, Image Gallery, Image Sprites

A).Opacity

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

Transparent Image

The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent:

```
img {  opacity: 0.5;}
```

Transparent Hover Effect

The opacity property is often used together with the :hover selector to change the opacity on mouse-over:

B).Navigation Bar

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:

```
<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>
```

a).Vertical Navigation Bar

```
li a {
  display: block;
  width: 60px;
}
```

Home
News
Contact
About

Example explained:

- display: block; - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width (and padding, margin, height, etc. if you want)
- width: 60px; - Block elements take up the full width available by default. We want to specify a 60 pixels width

Vertical Navigation Bar Examples

```
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;
}

/* Change the link color on hover */
li a:hover {
  background-color: #555;
  color: white;
}
```

Vertical Navigation Bar

Home
News
Contact
About

Active/Current Navigation Link

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

Center Links & Add Borders

Add text-align:center to <li> or <a> to center the links.

Add the border property to <ul> add a border around the navbar. If you also want borders inside the navbar, add a border-bottom to all <li> elements, except for the last one:

Full-height Fixed Vertical Navbar

Create a full-height, "sticky" side navigation:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  height: 100%; /* Full height */
  position: fixed; /* Make it stick, even on scroll */
  overflow: auto; /* Enable scrolling if the sidenav has
too much content */
}
```

b).Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar. Using inline or floating list items.

Inline List Items

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code from the previous page:

```
li {
  display: inline;
}
```

- display: inline; - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Floating List Items

Another way of creating a horizontal navigation bar is to float the <li> elements, and specify a layout for the navigation links:

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

li {
  float: left;
}

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

Home News Contact About

Note: If a !DOCTYPE is not specified, floating items can produce unexpected results.

A background color is added to the links to show the link area. The whole link area is clickable, not just the text.

Note: overflow:hidden is added to the ul element to prevent li elements from going outside of the list.

- float: left; - Use float to get block elements to float next to each other
- display: block; - Allows us to specify padding (and height, width, margins, etc. if you want)
- padding: 8px; - Specify some padding between each <a> element, to make them look good
- background-color: #dddddd; - Add a gray background-color to each <a> element

Tip: Add the background-color to <ul> instead of each <a> element if you want a full-width background color:

Horizontal Navigation Bar Examples

- Create a basic horizontal navigation bar with a dark background color and change the background color of the links when the user moves the mouse over them:

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

Home News Contact About

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

li a:hover {
  background-color: #111;
}
```

Active/Current Navigation Link

Right-Align Links

Right-align links by floating the list items to the right (float:right;):

```
<li style="float:right"><a class="active" href="#about">About</a></li>
```

Border Dividers

Add the border-right property to <li> to create link dividers:

Fixed Navigation Bar

```
position: fixed;
top: 0;
width: 100%;
```

```
position: fixed;
bottom: 0;
width: 100%;
```

Sticky Navbar

Add position: sticky; to <ul> to create a sticky navbar.

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).

```
ul {
  position: -webkit-sticky; /* Safari */
  position: sticky;
  top: 0;
}
```

Responsive

C).Dropdown

Basic Dropdown

Example Explained

HTML) Use any element to open the dropdown content, e.g. a <span>, or a <button> element.

Use a container element (like <div>) to create the dropdown content and add whatever you want inside of it.

Wrap a <div> element around the elements to position the dropdown content correctly with CSS.

CSS) The .dropdown class uses position:relative, which is needed when we want the dropdown content to be placed right below the dropdown button (using position:absolute).

The .dropdown-content class holds the actual dropdown content. It is hidden by default, and will be displayed on hover (see below). Note the min-width is set to 160px. Feel free to change this. **Tip:** If you want the width of the dropdown content to be as wide as the dropdown button, set the width to 100% (and overflow:auto to enable scroll on small screens).

Instead of using a border, we have used the CSS box-shadow property to make the dropdown menu look like a "card".

The :hover selector is used to show the dropdown menu when the user moves the mouse over the dropdown button.

Dropdown Menu

Create a dropdown menu that allows the user to choose an option from a list:

Dropdown Menu

This example is similar to the previous one, except that we add links inside the dropdown box and style them to fit a styled dropdown button:

`Click me`

Image Gallery

Click me

9).CSS Attribute Selectors, Forms ,Website layout,

A).Attribute Selectors

CSS [attribute] Selector

The [attribute] selector is used to select elements with a specified attribute.

The following example selects all <a> elements with a target attribute:

```
a[target] {
  background-color: yellow;
}
```

CSS [attribute="value"] Selector

The [attribute="value"] selector is used to select elements with a specified attribute and value.

The following example selects all <a> elements with a target="_blank" attribute:

```
a[target="_blank"] {
  background-color: yellow;
}
```

CSS [attribute~="value"] Selector

The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

```
<!DOCTYPE html>
<html>
<head>
<style>
[title~=flower] {
  border: 5px solid yellow;
}
</style>
</head>
<body>

<h2>CSS [attribute~="value"] Selector</h2>
<p>All images with the title attribute containing the word "flower" get a yellow border.</p>

<img src="klematis.jpg" title="klematis flower" width="150" height="113">
<img src="img_flwr.gif" title="flower" width="224" height="162">
<img src="img_tree.gif" title="tree" width="200" height="358">

</body>
</html>
```

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

CSS [attribute|="value"] Selector

The [attribute|="value"] selector is used to select elements with the specified attribute, whose value can be exactly the specified value, or the specified value followed by a hyphen (-).

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text".

```
 [class|="top"] {
 background: yellow;
}
```

CSS [attribute^="value"] Selector

The [attribute^="value"] selector is used to select elements with the specified attribute, whose value starts with the specified value.

The following example selects all elements with a class attribute value that starts with "top":

Note: The value does not have to be a whole word!

```
 [class^="top"] {
 background: yellow;
}
```

CSS [attribute$="value"] Selector

The [attribute$="value"] selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not have to be a whole word!

```
 [class$="test"] {
 background: yellow;
}
```

CSS [attribute*="value"] Selector

The [attribute*="value"] selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

```
 [class*="te"] {
 background: yellow;
}
```

Click me

B).Forms

Click me

Click me

10).CSS Units, Specificity, !important Rule, Math Functions

A).Units

CSS has several different units for expressing a length.

Many CSS properties take "length" values, such as width, margin, padding, font-size, etc.

Length is a number followed by a length unit, such as 10px, 2em, etc.

```
h1 {
  font-size: 60px;
}

p {
  font-size: 25px;
  line-height: 50px;
}
```

Note: A whitespace cannot appear between the number and the unit. However, if the value is 0, the unit can be omitted.

For some CSS properties, negative lengths are allowed.

There are two types of length units: **absolute** and **relative**.

Absolute Lengths

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

Unit	**Description**
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in
pc	picas (1pc = 12 pt)

* Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.

Relative Lengths

Relative length units specify a length relative to another length property. Relative length units scales better between different rendering mediums.

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

B).Specificity

What is Specificity?

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

Think of specificity as a score/rank that determines which style declaration are ultimately applied to an element.

Look at the following examples:

- In this example, we have added a class selector (named "test"), and specified a green color for this class. The text will now be green (even though we have specified a red color for the element selector "p". This is because the class selector is given higher priority:

- In this example, we have added the id selector (named "demo"). The text will now be blue, because the id selector is given higher priority:

- In this example, we have added an inline style for the "p" element. The text will now be pink, because the inline style is given the highest priority:

Secificity Hierarchy

Every CSS selector has its place in the specificity hierarchy.

There are four categories which define the specificity level of a selector:

- **Inline styles** - Example: <h1 style="color: pink;">
- **IDs** - Example: #navbar
- **Classes, pseudo-classes, attribute selectors** - Example: .test, :hover, [href]
- **Elements and pseudo-elements** - Example: h1, :before

C)!important Rule

What is !important?

The !important rule in CSS is used to add more importance to a property/value than normal.

In fact, if you use the !important rule, it will override ALL previous styling rules for that specific property on that element!

Let us look at an example:

```
#myid {
  background-color: blue;
}

.myclass {
  background-color: gray;
}

p {
  background-color: red !important;
}
```

D).Math Functions

The CSS math functions allow mathematical expressions to be used as property values. Here, we will explain the calc(), max() and min() functions.

The calc() Function

The calc() function performs a calculation to be used as the property value.

```
width: calc(100% - 100px);
```

The max() Function

The max() function uses the largest value, from a comma-separated list of values, as the property value.

```
width: max(50%, 300px);
```

The min() Function

The min() function uses the smallest value, from a comma-separated list of values, as the property value.

```
width: min(50%, 300px);
```

Advanced

1).CSS Rounded Corners, Border Images, Multiple Backgrounds, Colors, Color Keywords, Gradients,Shadow Effects, Text Effects

A)Rounded Corners

```
border-radius
```

B) Border Images

```
border: 10px solid transparent;
  padding: 15px;
  border-image: url(border.png) 30 round;
}
```

C) Multiple Backgrounds

- `background-size`
- `background-origin`
- `background-clip`

```
background-image: url(img_flwr.gif), url(paper.gif);
  background-position: right bottom, left top;
  background-repeat: no-repeat, repeat;
```

Multiple background images can be specified using either the individual background properties (as above) or the background shorthand property.

The following example uses the background shorthand property (same result as example above):

```
  background: url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;
```

The CSS background-size property allows you to specify the size of background images.

The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

The following example resizes a background image to much smaller than the original image (using pixels):

```
background: url(img_flower.jpg);
  background-size: 100px 80px;
  background-repeat: no-repeat;
```

The two other possible values for background-size are contain and cover.

The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there may be some areas of the background which are not covered by the background image.

The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area.

The following example illustrates the use of contain and cover:

```
  background: url(img_flower.jpg);
  background-size: contain;
  background-repeat: no-repeat;
```

```
  background: url(img_flower.jpg);
  background-size: cover;
  background-repeat: no-repeat;
```

Hero Image

Click Me

CSS background-origin Property

The CSS background-origin property specifies where the background image is positioned.

The property takes three different values:

- border-box - the background image starts from the upper left corner of the border
- padding-box - (default) the background image starts from the upper left corner of the padding edge
- content-box - the background image starts from the upper left corner of the content

The following example illustrates the background-origin property:

CSS background-clip Property

The CSS background-clip property specifies the painting area of the background.

The property takes three different values:

- border-box - (default) the background is painted to the outside edge of the border
- padding-box - the background is painted to the outside edge of the padding
- content-box - the background is painted within the content box

The following example illustrates the background-clip property:

D)Colors

This page will explain the transparent, currentcolor, and inherit keywords.

The transparent Keyword

```
body {
  background-image: url("paper.gif");
}

div {
  background-color: transparent;
}
```

The currentcolor Keyword

The currentcolor keyword is like a variable that holds the current value of the color property of an element.

This keyword can be useful if you want a specific color to be consistent in an element or a page.

```
div {
  color: blue;
  border: 10px solid currentcolor;
}

body {
  color: purple;
}

div {
  background-color: currentcolor;
}
```

The inherit Keyword

The inherit keyword specifies that a property should inherit its value from its parent element.

The inherit keyword can be used for any CSS property, and on any HTML element.

```
div {
  border: 2px solid red;
}

span {
  border: inherit;
}
```

E).Gradients

Linear Gradients

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
background-image: repeating-linear-gradient(red, yellow 10%, green 20%);
```

Radial Gradients

```
background-image: radial-gradient(shape size at position, start-color, ..., last-color);
background-image: radial-gradient(circle, red, yellow, green);
background-image: repeating-radial-gradient(red, yellow 10%, green 15%);
```

Conic Gradients

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

F).Shadow Effects

Text Shadow

Box Shadow

G). Text Effects

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

text-overflow: clip;
text-overflow: ellipsis;

word-wrap: break-word;
word-break: keep-all;
word-break: break-all;

```
<!DOCTYPE html>
<html>
<head>
<style>
p.test1 {
  writing-mode: horizontal-tb;
}

span.test2 {
  writing-mode: vertical-rl;
}

p.test2 {
  writing-mode: vertical-rl;
}
</style>
</head>
<body>
<h1>The writing-mode Property</h1>

<p class="test1">Some text with default writing-mode.</p>

<p>Some text with a span element with a <span
class="test2">vertical-rl</span> writing-mode.</p>

<p class="test2">Some text with writing-mode: vertical-rl.</p>

</body>
</html>
```

US VS Cord

2).CSS Web Fonts, 2D Transforms, 3D Transforms, Transitions

A).Web Fonts

@font-face Rule

Web fonts allow Web designers to use fonts that are not installed on the user's computer.

When you have found/bought the font you wish to use, just include the font file on your web server, and it will be automatically downloaded to the user when needed.

Your "own" fonts are defined within the CSS @font-face rule.

Different Font Formats

TrueType Fonts (TTF)

TrueType is a font standard developed in the late 1980s, by Apple and Microsoft. TrueType is the most common font format for both the Mac OS and Microsoft Windows operating systems.

OpenType Fonts (OTF)

OpenType is a format for scalable computer fonts. It was built on TrueType, and is a registered trademark of Microsoft. OpenType fonts are used commonly today on the major computer platforms.

The Web Open Font Format (WOFF)

WOFF is a font format for use in web pages. It was developed in 2009, and is now a W3C Recommendation. WOFF is essentially OpenType or TrueType with compression and additional metadata. The goal is to support font distribution from a server to a client over a network with bandwidth constraints.

The Web Open Font Format (WOFF 2.0)

TrueType/OpenType font that provides better compression than WOFF 1.0.

SVG Fonts/Shapes

SVG fonts allow SVG to be used as glyphs when displaying text. The SVG 1.1 specification define a font module that allows the creation of fonts within an SVG document. You can also apply CSS to SVG documents, and the @font-face rule can be applied to text in SVG documents.

Embedded OpenType Fonts (EOT)

EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

Using The Font You Want

In the @font-face rule; first define a name for the font (e.g. myFirstFont) and then point to the font file.

To use the font for an HTML element, refer to the name of the font (myFirstFont) through the font-family property:

```
@font-face {
  font-family: myFirstFont;
  src: url(sansation_light.woff);
}

div {
  font-family: myFirstFont;
}
```

Using Bold Text

You must add another @font-face rule containing descriptors for bold text:

```
@font-face {
  font-family: myFirstFont;
  src: url(sansation_bold.woff);
  font-weight: bold;
}
```

The file "sansation_bold.woff" is another font file, that contains the bold characters for the Sansation font.

Browsers will use this whenever a piece of text with the font-family "myFirstFont" should render as bold.

This way you can have many @font-face rules for the same font.

B).2D Transforms

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

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

In this chapter you will learn about the following CSS property:

- transform

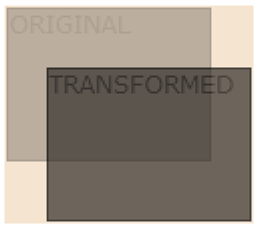
The numbers in the table specify the first browser version that fully supports the property.

2D Transforms Methods

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

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

translate() Method

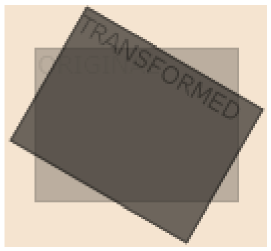


The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:

```
div {
  transform: translate(50px, 100px);
}
```

The rotate() Method



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

The following example rotates the <div> element clockwise with 20 degrees:

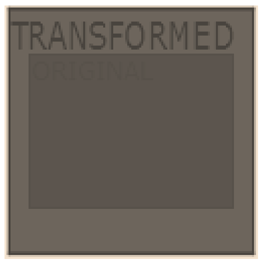
```
div {
  transform: rotate(20deg);
}
```

Using negative values will rotate the element counter-clockwise.

The following example rotates the <div> element counter-clockwise with 20 degrees:

```
div {
  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).

The following example increases the <div> element to be two times of its original width, and three times of its original height:

```
div {
  transform: scale(2, 3);
}
```

The following example decreases the <div> element to be half of its original width and height:

```
div {
  transform: scale(0.5, 0.5);
}
```

The scaleX() Method

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

The following example increases the <div> element to be two times of its original width:

```
div {
  transform: scaleX(2);
}
```

The following example decreases the <div> element to be half of its original width:

```
div {
  transform: scaleX(0.5);
}
```

The scaleY() Method

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

The following example increases the <div> element to be three times of its original height:

```
div {
  transform: scaleY(3);
}
```

The following example decreases the <div> element to be half of its original height:

```
div {
  transform: scaleY(0.5);
}
```

The skewX() Method

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

The following example skews the <div> element 20 degrees along the X-axis:

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

The skewY() Method

The skewY() method skews an element along the Y-axis by the given angle.

The following example skews the <div> element 20 degrees along the Y-axis:

```
div {
  transform: skewY(20deg);
}
```

The skew() Method

The skew() method skews an element along the X and Y-axis by the given angles.

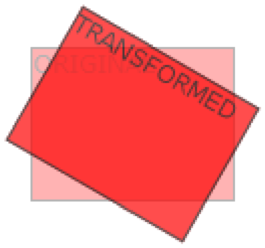
The following example skews the <div> element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

```
div {
  transform: skew(20deg, 10deg);
}
```

If the second parameter is not specified, it has a zero value. So, the following example skews the <div> element 20 degrees along the X-axis:

```
div {
  transform: skew(20deg);
}
```

The matrix() Method



The matrix() method combines all the 2D transform methods into one.

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow: matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

```
div {
  transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```

C).3D Transforms

CSS also supports 3D transformations.

Mouse over the elements below to see the difference between a 2D and a 3D transformation:

In this chapter you will learn about the following CSS property:

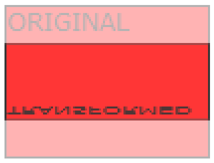
- transform

3D Transforms Methods

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

- rotateX()
- rotateY()
- rotateZ()

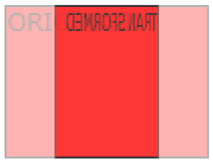
The rotateX() Method



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

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

The rotateY() Method



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

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

The rotateZ() Method

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

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

D).CSS Transitions

CSS Transitions

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

Mouse over the element below to see a CSS transition effect:

In this chapter you will learn about the following properties:

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

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:

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

The transition effect will start when the specified CSS property (width) changes value.

Now, let us specify a new value for the width property when a user mouses over the <div> element:

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

Notice that when the cursor mouses out of the element, it will gradually change back to its original style.

Change Several Property Values

The following example 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 {
  transition: width 2s, height 4s;
}
```

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

The following example shows some of the different speed curves that can be used:

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

Delay the Transition Effect

- The transition-delay property specifies a delay (in seconds) for the transition effect.
- The following example has a 1 second delay before starting:

```
div {
  transition-delay: 1s;
}
```

Transition + Transformation

- The following example adds a transition effect to the transformation:

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

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

More Transition Examples

The CSS transition properties can be specified one by one, like this:

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

or by using the shorthand property transition:

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

3).CSS Animations, Tooltip, Styling Images, Image Reflection, object-fit Property, object-position Property

A)Animations

In this chapter you will learn about the following properties:

- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

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.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

```
/* The animation code */
@keyframes example {
 from {background-color: red;}
 to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
 width: 100px;
 height: 100px;
 background-color: red;
 animation-name: example;
animation-duration: 4s;
}
```

Note: The animation-duration property defines how long an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
/* The animation code */
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

The following example will change both the background-color and the position of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
/* The animation code */
@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;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

Delay an Animation

The animation-delay property specifies a delay for the start of an animation.

The following example has a 2 seconds delay before starting the animation:

```
div {
 width: 100px;
 height: 100px;
 position: relative;
 background-color: red;
 animation-name: example;
 animation-duration: 4s;
 animation-delay: 2s;
}
```

Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for *N* seconds.

In the following example, the animation will start as if it had already been playing for 2 seconds:

```
div {
 width: 100px;
 height: 100px;
 position: relative;
 background-color: red;
 animation-name: example;
 animation-duration: 4s;
 animation-delay: -2s;
}
```

Set How Many Times an Animation Should Run

The animation-iteration-count property specifies the number of times an animation should run.

The following example will run the animation 3 times before it stops:

```
div {
 width: 100px;
 height: 100px;
 position: relative;
 background-color: red;
 animation-name: example;
 animation-duration: 4s;
 animation-iteration-count: 3;
}
```

The following example uses the value "infinite" to make the animation continue for ever:

```
div {
 width: 100px;
 height: 100px;
 position: relative;
 background-color: red;
 animation-name: example;
 animation-duration: 4s;
 animation-iteration-count: infinite;
}
```

Run Animation in Reverse Direction or Alternate Cycles

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

The following example will run the animation in reverse direction (backwards):

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-direction: reverse;
}
```

The following example uses the value "alternate" to make the animation run forwards first, then backwards:

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-iteration-count: 2;
  animation-direction: alternate;
}
```

The following example uses the value "alternate-reverse" to make the animation run backwards first, then forwards:

Specify the Speed Curve of the Animation

The animation-timing-function property specifies the speed curve of the animation.

The animation-timing-function property can have the following values:

- ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
- linear - Specifies an animation with the same speed from start to end
- ease-in - Specifies an animation with a slow start
- ease-out - Specifies an animation with a slow end
- ease-in-out - Specifies an animation with a slow start and end
- cubic-bezier(n,n,n,n) - Lets you define your own values in a cubic-bezier function

The following example shows some of the different speed curves that can be used:

```
#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;}
```

Click

Animation Shorthand Property

The example below uses six of the animation properties:

```
div {
  animation-name: example;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

The same animation effect as above can be achieved by using the shorthand animation property:

```
div {
  animation: example 5s linear 2s infinite alternate;
}
```

B).Tooltip

Basic Tooltip

Create a tooltip that appears when the user moves the mouse over an element:

```
/* Tooltip container */
.tooltip {
  position: relative;
  display: inline-block;
  border-bottom: 1px dotted black; /* If you want dots under the hoverable text */
}

/* Tooltip text */
.tooltip .tooltiptext {
  visibility: hidden;
  width: 120px;
  background-color: black;
  color: #fff;
  text-align: center;
  padding: 5px 0;
  border-radius: 6px;

  /* Position the tooltip text - see examples below! */
  position: absolute;
  z-index: 1;
}

/* Show the tooltip text when you mouse over the tooltip container */
.tooltip:hover .tooltiptext {
  visibility: visible;
}
</style>
```

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

Example Explained

HTML: Use a container element (like <div>) and add the "tooltip" class to it. When the user mouse over this <div>, it will show the tooltip text.

The tooltip text is placed inside an inline element (like <span>) with class="tooltiptext".

CSS: The tooltip class use position:relative, which is needed to position the tooltip text (position:absolute). **Note:** See examples below on how to position the tooltip.

The tooltiptext class holds the actual tooltip text. It is hidden by default, and will be visible on hover (see below). We have also added some basic styles to it: 120px width, black background color, white text color, centered text, and 5px top and bottom padding.

The CSS border-radius property is used to add rounded corners to the tooltip text.

The :hover selector is used to show the tooltip text when the user moves the mouse over the <div> with class="tooltip".

Positioning Tooltips

In this example, the tooltip is placed to the right (left:105%) of the "hoverable" text (<div>). Also note that top:-5px is used to place it in the middle of its container element. We use the number 5 because the tooltip text has a top and bottom padding of 5px. If you increase its padding, also increase the value of the top property to ensure that it stays in the middle (if this is something you want). The same applies if you want the tooltip placed to the left.

Right Tooltip

```
.tooltip .tooltiptext {
  top: -5px;
  left: 105%;
}
```

Left Tooltip

```
.tooltip .tooltiptext {
  top: -5px;
  right: 105%;
}
```

If you want the tooltip to appear on top or on the bottom, see examples below. Note that we use the margin-left property with a value of minus 60 pixels. This is to center the tooltip above/below the hoverable text. It is set to the half of the tooltip's width (120/2 = 60).

Top Tooltip

```
.tooltip .tooltiptext {
  width: 120px;
  bottom: 100%;
  left: 50%;
  margin-left: -60px; /* Use half of the width (120/2 = 60), to center the tooltip */
}
```

Bottom Tooltip

```
.tooltip .tooltiptext {
  width: 120px;
```

```
 top: 100%;
 left: 50%;
 margin-left: -60px; /* Use half of the width (120/2 = 60), to center the tooltip */
}
```

Tooltip Arrows

To create an arrow that should appear from a specific side of the tooltip, add "empty" content after tooltip, with the pseudo-element class ::after together with the content property. The arrow itself is created using borders. This will make the tooltip look like a speech bubble.

This example demonstrates how to add an arrow to the bottom of the tooltip:

Bottom Arrow

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

Example Explained

Position the arrow inside the tooltip: top: 100% will place the arrow at the bottom of the tooltip. left: 50% will center the arrow.

Note: The border-width property specifies the size of the arrow. If you change this, also change the margin-left value to the same. This will keep the arrow centered.

The border-color is used to transform the content into an arrow. We set the top border to black, and the rest to transparent. If all sides were black, you would end up with a black square box.

This example demonstrates how to add an arrow to the top of the tooltip. Notice that we set the bottom border color this time:

Fade In Tooltips (Animation)

If you want to fade in the tooltip text when it is about to be visible, you can use the CSS transition property together with the opacity property, and go from being completely invisible to 100% visible, in a number of specified seconds (1 second in our example):

```
.tooltip .tooltiptext {
 opacity: 0;
 transition: opacity 1s;
}

.tooltip:hover .tooltiptext {
 opacity: 1;
}
```

C).Styling Images

Use the border-radius property to create rounded images:

Thumbnail Images

Use the border property to create thumbnail images.

Thumbnail Image: Border

Thumbnail Image as Link:

Responsive Images

Responsive images will automatically adjust to fit the size of the screen.

Resize the browser window to see the effect:

If you want an image to scale down if it has to, but never scale up to be larger than its original size, add the following:

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

Center an Image

To center an image, set left and right margin to auto and make it into a block element:

```
img {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 50%;
}
```

Image Hover Overlay

Fade in text: Click Me

Fade in a box: Click me

Slide in (top): Click me

Slide in (bottom): Click me

Slide in (left): Click me

Slide in (right): Click me

Flip an Image

```
img:hover {
  transform: scaleX(-1);
}
```

Responsive Image Gallery

CSS can be used to create image galleries. This example use media queries to re-arrange the images on different screen sizes. Resize the browser window to see the effect:

Click me

D).Image Reflection

CSS Image Reflections

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

The value of the box-reflect property can be: below, above, left , or right.

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

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

CSS Reflection Offset

To specify the gap between the image and the reflection, add the size of the gap to the box-reflect property.

Here we want the reflection below the image, with a 20px offset:

```
img {
  -webkit-box-reflect: below 20px;
}
```

CSS Reflection With Gradient

We can also create a fade-out effect on the reflection.

Create a fade-out effect on the reflection:

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

E).object-fit Property

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

This property tells the content to fill the container in a variety of ways; such as "preserve that aspect ratio" or "stretch up and take up as much space as possible".

Look at the following image from Paris. This image is 400 pixels wide and 300 pixels high:

We see that the image is being squished to fit the container of 200x300 pixels (its original aspect ratio is destroyed).

Here is where the 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

Using object-fit: cover;

If we use object-fit: cover; the image keeps its aspect ratio and fills the given dimension. The image will be clipped to fit:

```
img {
  width: 200px;
  height: 300px;
  object-fit: cover;
}
```

Using object-fit: contain;

If we use object-fit: contain; the image keeps its aspect ratio, but is resized to fit within the given dimension:

```
img {
  width: 200px;
  height: 300px;
  object-fit: contain;
}
```

Using object-fit: fill;

If we use object-fit: fill; the image is resized to fill the given dimension. If necessary, the image will be stretched or squished to fit:

```
img {
  width: 200px;
  height: 300px;
  object-fit: fill;
}
```

Using object-fit: none;

If we use object-fit: none; the image is not resized:

```
img {
  width: 200px;
  height: 300px;
  object-fit: none;
}
```

Using object-fit: scale-down;

If we use object-fit: scale-down; the image is scaled down to the smallest version of none or contain:

```
img {
  width: 200px;
  height: 300px;
  object-fit: scale-down;
}
```

click me

F).object-position Property

Using the object-position Property

Let's say that the part of the image that is shown, is not positioned as we want. To position the image, we will use the object-position property.

Here we will use the object-position property to position the image so that the great old building is in center:

```
img {
  width: 200px;
  height: 300px;
  object-fit: cover;
  object-position: 80% 100%;
}
```

Here we will use the object-position property to position the image so that the famous Eiffel Tower is in center:

```
img {
  width: 200px;
  height: 300px;
  object-fit: cover;
  object-position: 15% 100%;
}
```

4).CSS Masking, Buttons, Pagination, Multiple Columns , User Interface

A).mask-image Property

The CSS mask-image property specifies a mask layer image.

The mask layer image can be a PNG image, an SVG image, a CSS gradient, or an SVG <mask> element.

Use an Image as the Mask Layer

To use a PNG or an SVG image as the mask layer, use a url() value to pass in the mask layer image.

The mask image needs to have a transparent or semi-transparent area. Black indicates fully transparent.

Here is the mask image (a PNG image) we will use:

```
.mask1 {
 -webkit-mask-image: url();
 mask-image: url();
 -webkit-mask-repeat: no-repeat;
 mask-repeat: no-repeat;
}
```

Example Explained

The mask-image property specifies the image to be used as a mask layer for an element.

The mask-repeat property specifies if or how a mask image will be repeated. The no-repeat value indicates that the mask image will not be repeated (the mask image will only be shown once).

```
.mask1 {
 -webkit-mask-image: url(.png);
 mask-image: url(.png);
}
```

Use Gradients as the Mask Layer

CSS linear and radial gradients can also be used as mask images.

Linear Gradient Examples

Here, we use a linear-gradient as the mask layer for our image. This linear gradient goes from top (black) to bottom (transparent):

Use a linear gradient as a mask layer:

```
.mask1 {
  -webkit-mask-image: linear-gradient(black, transparent);
  mask-image: linear-gradient(black, transparent);
}
```

Use a linear gradient along with text masking as a mask layer:

```
.mask1 {
  max-width: 600px;
  height: 350px;
  overflow-y: scroll;
  background: url(img_5terre.jpg) no-repeat;
  -webkit-mask-image: linear-gradient(black, transparent);
  mask-image: linear-gradient (black, transparent);
}
```

Radial Gradient Examples

Here, we use a radial-gradient (shaped as a circle) as the mask layer for our image:

Use a radial gradient as a mask layer (a circle):

```
.mask2 {
  -webkit-mask-image: radial-gradient(circle, black 50%, rgba(0, 0, 0, 0.5) 50%);
  mask-image: radial-gradient(circle, black 50%, rgba(0, 0, 0, 0.5) 50%);
}
```

Here, we use a radial-gradient (shaped as an ellipse) as the mask layer for our image:

Use another radial gradient as a mask layer (an ellipse):

```
.mask3 {
  -webkit-mask-image: radial-gradient(ellipse, black 50%, rgba(0, 0, 0, 0.5) 50%);
  mask-image: radial-gradient(ellipse, black 50%, rgba(0, 0, 0, 0.5));
}
```

Use SVG as the Mask Layer

The SVG <mask> element can be used inside an SVG graphic to create masking effects.

Here, we use the SVG <mask> element to create different mask layers for our image:

An SVG mask layer (formed as a triangle):

```
<svg width="600" height="400">
 <mask id="svgmask1">
  <polygon fill="#ffffff" points="200 0, 400 400, 0 400"></polygon>
 </mask>
 <image xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="img_5terre.jpg" mask="url(#svgmask1)"></image>
</svg>
```

An SVG mask layer (formed as a star):

```
<svg width="600" height="400">
 <mask id="svgmask2">
  <polygon fill="#ffffff" points="100,10 40,198 190,78 10,78 160,198"></polygon>
 </mask>
 <image xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="img_5terre.jpg" mask="url(#svgmask2)"></image>
</svg>
```

An SVG mask layer (formed as circles):

```
<svg width="600" height="400">
 <mask id="svgmask3">
  <circle fill="#ffffff" cx="75" cy="75" r="75"></circle>
  <circle fill="#ffffff" cx="80" cy="260" r="75"></circle>
  <circle fill="#ffffff" cx="270" cy="160" r="75"></circle>
 </mask>
 <image xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="img_5terre.jpg" mask="url(#svgmask3)"></image>
</svg>
```

B).Buttons

Click me

C). Pagination

```
.pagination {
  display: inline-block;
}

.pagination a {
  color: black;
  float: left;
  padding: 8px 16px;
  text-decoration: none;
}
```

Active and Hoverable Pagination

Highlight the current page with an .active class, and use the :hover selector to change the color of each page link when moving the mouse over them:

```
.pagination a.active {
  background-color: #4CAF50;
  color: white;
}

.pagination a:hover:not(.active) {background-color: #ddd;}
```

Rounded Active and Hoverable Buttons

Add the border-radius property if you want a rounded "active" and "hover" button:

```
.pagination a {
  border-radius: 5px;
}

.pagination a.active {
  border-radius: 5px;
}
```

Hoverable Transition Effect

Add the transition property to the page links to create a transition effect on hover:

```
.pagination a {
  transition: background-color .3s;
}
```

D).Multiple Columns

CSS Multi-column Properties

In this chapter you will learn about the following multi-column properties:

- column-count
- column-gap
- column-rule-style
- column-rule-width
- column-rule-color
- column-rule
- column-span
- column-width

CSS Create Multiple Columns

The column-count property specifies the number of columns an element should be divided into.

The following example will divide the text in the <div> element into 3 columns:

```
div {
  column-count: 3;
}
```

CSS Specify the Gap Between Columns

The column-gap property specifies the gap between the columns.

The following example specifies a 40 pixels gap between the columns:

```
div {
  column-gap: 40px;
}
```

CSS Column Rules

The column-rule-style property specifies the style of the rule between columns:

```
div {
  column-rule-style: solid;
}
```

The column-rule-width property specifies the width of the rule between columns:

```
div {
  column-rule-width: 1px;
}
```

The column-rule-color property specifies the color of the rule between columns:

```
div {
  column-rule-color: lightblue;
}
```

The column-rule property is a shorthand property for setting all the column-rule-* properties above.

The following example sets the width, style, and color of the rule between columns:

```
div {
  column-rule: 1px solid lightblue;
}
```

Specify How Many Columns an Element Should Span

The column-span property specifies how many columns an element should span across.

The following example specifies that the <h2> element should span across all columns:

```
h2 {
  column-span: all;
}
```

Specify The Column Width

The column-width property specifies a suggested, optimal width for the columns.

The following example specifies that the suggested, optimal width for the columns should be 100px:

```
div {
  column-width: 100px;
}
```

E).User Interface

In this chapter you will learn about the following CSS user interface properties:

- resize
- outline-offset

CSS Resizing

The resize property specifies if (and how) an element should be resizable by the user.

This div element is resizable by the user!

To resize: Click and drag the bottom right corner of this div element.

The following example lets the user resize only the width of a <div> element:

```
div {
  resize: horizontal;
  overflow: auto;
}
```

The following example lets the user resize only the height of a <div> element:

```
div {
  resize: vertical;
  overflow: auto;
}
```

The following example lets the user resize both the height and width of a <div> element:

```
div {
  resize: both;
  overflow: auto;
}
```

```
textarea {
  resize: none;
}
```

5).CSS Media Queries, Media Queries - Examples, Flexbox, User Interface

A).Media Queries

CSS2 Introduced Media Types

The @media rule, introduced in CSS2, made it possible to define different style rules for different media types.

Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.

Unfortunately these media types never got a lot of support by devices, other than the print media type.

CSS3 Introduced Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

- Media queries can be used to check many things, such as:
- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

Media Query Syntax

A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

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

The result of the query is true if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules.

Unless you use the not or only operators, the media type is optional and the all type will be implied.

You can also have different stylesheets for different media:

<link rel="stylesheet" media="*mediatype* and|not|only (*expressions*)" href="*print.css*">

CSS3 Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

Media Queries Simple Examples

One way to use media queries is to have an alternate CSS section right inside your style sheet.

The following example changes the background-color to lightgreen if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the background-color will be pink):

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

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

Click me

B).More Examples

- Let us look at some more examples of using media queries.
- Media queries are a popular technique for delivering a tailored style sheet to different devices. To demonstrate a simple example, we can change the background color for different devices:

```
@media screen and (max-width: 767px){
    body{background-color: aqua;
    }
}
@media screen and (min-width: 768px) and (max-width: 1024px) {
    body{
        background-color: aqua;
    }
   }
```

Media Queries For Menus

- In this example, we use media queries to create a responsive navigation menu, that varies in design on different screen sizes.

```
/* The navbar container */
.topnav {
  overflow: hidden;
  background-color: #333;
}

/* Navbar links */
.topnav a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}
```

```
/* On screens that are 600px wide or less, make the menu links stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
  .topnav a {
    float: none;
    width: 100%;
  }
}
```

Media Queries For Columns

A common use of media queries, is to create a flexible layout. In this example, we create a layout that varies between four, two and full-width columns, depending on different screen sizes:

Large screens:

Medium screens:

Small screens:

click me

Click me

Flexible Image Gallery

In this example, we use media queries together with flexbox to create a responsive image gallery:

Clicke me

Flexible Website

In this example, we use media queries together with flexbox to create a responsive website, containing a flexible navigation bar and flexible content.

Click me

Min Width to Max Width

You can also use the (max-width: ..) and (min-width: ..) values to set a minimum width and a maximum width.

For example, when the browser's width is between 600 and 900px, change the appearance of a <div> element:

```
@media screen and (max-width: 900px) and (min-width: 600px) {
  div.example {
    font-size: 50px;
    padding: 50px;
    border: 8px solid black;
    background: yellow;
  }
}
```

```
/* When the width is between 600px and 900px OR above 1100px - change the appearance of <div> */
@media screen and (max-width: 900px) and (min-width: 600px), (min-width: 1100px) {
  div.example {
    font-size: 50px;
    padding: 50px;
    border: 8px solid black;
    background: yellow;
  }
}
```

Clicke me

Videos For More Info.

CSS End Best Of Luck For JavaScript