

# CSS

Notes by [rajeeshvengalapurath@gmail.com](mailto:rajeeshvengalapurath@gmail.com)

## What is CSS?

CSS is **not** a programming language. It's **not** a markup language either.  
**CSS is a style sheet language.**

## How to Use

### Inline - Style attribute inside HTML elements

```
<h1 style="color:blue;">A Blue Heading</h1>
```

### Internal - <style> element in the <head> section

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {background-color: powderblue;}
      h1   {color: blue;}
      p    {color: red;}
    </style>
  </head>
  <body>

    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>

  </body>
</html>
```

### External - <link> element to link to an external CSS file

#### Html File

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

## Styles.css File

```
body {  
    background-color: powderblue;  
}  
h1 {  
    color: blue;  
}  
p {  
    color: red;  
}
```

## Viewport

The viewport is the user's visible area of a web page. The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

## Syntax

```
Selector  
↓                               ← Declaration →  
h1 {color: blue; font-size: 12px}  
    ↑       ↑  
    Property Value
```

## Simple Selectors

Select elements based on name, id, class

### Element Selector

```
p { text-align: center; color: red; }
```

### Id Selector

```
#para1 { text-align: center; color: red; }
```

### Class Selector

```
.center { text-align: center; color: red; }  
p.center { text-align: center; color: red; }
```

```
/*<p> element will be styled according to class="center" and to class="large"*/  
<p class="center large">This paragraph refers to two classes.</p>
```

### Universal Selector

```
* { text-align: center; color: blue; }
```

# Combinator Selectors

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

## Descendant Selector

Matches all elements that are descendants of a specified element. (hierarchical)

```
.rootdiv p {color: red;}
```

```
<div class="rootdiv">
  <p>div1</p>
  <p>div1</p>
  <div>
    <p>div2</p>
    <p>div2</p>
  </div>
</div>
<p>out1</p>
<p>out2</p>
```

## Child Selector

Selects all elements that are the children of a specified element. (Not child's child)

```
.rootdiv > p {color: red;}
```

```
<div class="rootdiv">
  <p>div1</p>
  <p>div1</p>
  <div>
    <p>div2</p>
    <p>div2</p>
  </div>
</div>
<p>out1</p>
<p>out2</p>
```

## Adjacent Sibling Selector

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

```
.rootdiv + p {color: red;}
```

```
<div class="rootdiv">
  <p>div1</p>
  <p>div1</p>
  <div>
    <p>div2</p>
    <p>div2</p>
  </div>
</div>
<p>out1</p>
```

<p>out2</p>

## General Sibling Selector

Selects all elements that are siblings of a specified element.

```
.rootdiv ~ p {color: red;}
```

```
<div class="rootdiv">
  <p>div1</p>
  <p>div1</p>
  <div>
    <p>div2</p>
    <p>div2</p>
  </div>
</div>
<p>out1</p>
<p>out2</p>
```

## All CSS Combinator Selectors

Selector	Example	Example description
element element	div p	Selects all <p> elements inside <div> elements
element>element	div > p	Selects all <p> elements where the parent is a <div> element
element+element	div + p	Selects all <p> elements that are placed immediately after <div> elements
element1~element2	p ~ ul	Selects every <ul> element that are preceded by a <p> element

## Pseudo-classes

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

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

## Syntax

```
selector:pseudo-class { property: value; }
```

```
a:link { color: #FF0000; } /* unvisited link */
a:visited { color: #00FF00; } /* visited link */
a:hover { color: #FF00FF; } /* mouse over link */
```

```

a:active { color: #0000FF; } /* selected link */
div:hover { background-color: blue; } /* Hover on <div> */
p:first-child { color: blue; } /*A specified element that is the first child of
                                another element.*/
a.highlight:hover { color: #ff0000; } /*Pseudo-classes combined with CSS classes*/
p i:first-child { color: blue; } /*First <i> element in all <p> elements */
p:first-child i { color: blue; } /*All <i> elements in all first child <p>
                                elements*/

```

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

## Show div on Hover the p else Hide

```

p { display: none; background-color: yellow; padding: 20px; }
div:hover p { display: block; }

```

## All CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(language)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"

:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

# Pseudo-elements

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

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

## Syntax

```
selector::pseudo-element { property: value; }  
p::first-line { color: #ff0000; font-variant: small-caps; }  
  
/*Matches the portion of an element that is selected by a user*/  
p::selection { color: red; background: yellow; }
```

## Pseudo-elements + CSS classes

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

## Pseudo-elements + Pseudo-elements

```
p::first-letter { color: #ff0000; font-size: xx-large; }  
p::first-line { color: #0000ff; font-variant: small-caps; }
```

The first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color

## All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

# Attribute Selectors

Style HTML Elements With Specific Attributes

```
/*"target" as the attribute name*/  
a[target] { background-color: yellow; }
```

```

/*with a particular word*/
[title~="flower"] { border: 5px solid yellow; }

/*starting with the specified value.*/
[class|="top"] { background: yellow; }

/*attribute value begins with a specified value*/
[class^="top"] { background: yellow; }

/*attribute value ends with a specified value*/
[class$="test"] { background: yellow; }

/*attribute value contains a specified value*/
[class*="te"] { background: yellow; }

```

## Styling Forms

The attribute selectors can be useful for styling forms without class or ID

```

input[type="text"] { width: 150px; display: block; margin-bottom: 10px;
    background-color: yellow; }

input[type="button"] { width: 120px; margin-left: 35px; display: block; }

```

## All CSS Attribute Selectors

Selector	Example	Example description
[attribute]	[target]	Selects all elements with a target attribute
[attribute=value]	[target=_blank]	Selects all elements with target="_blank"
[attribute~=value]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[attribute =value]	[lang=en]	Selects all elements with a lang attribute value starting with "en"
[attribute^=value]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[attribute\$=value]	a[href\$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[attribute*=value]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"

## ::after

Inserts something after the content of each selected element(s)

```
/*An element like thing is added after the div with attributes mentioned below*/
div::after{ content: "myafter"; display: inline-block; height: 100px;
background-color: red; }
```

## Grouping Selector

Instead of

```
h1 { text-align: center; color: red; }
h2 { text-align: center; color: red; }
p { text-align: center; color: red; }
```

Can group like this

```
h1, h2, p { text-align: center; color: red; }
```

## Colors

### Background Color

```
background-color:DodgerBlue;
```

### Text Color

```
color:DodgerBlue;
```

### Border Color

```
border:2px solid Tomato;
```

### Color Values

```
background-color:rgb(255, 99, 71);
background-color:#ff6347;
background-color:hsl(9, 100%, 64%);
```

## Backgrounds

### Background Color

```
background-color:DodgerBlue;
```

### Opacity / Transparency

```
background-color: green; opacity: 0.3;
```

## Transparency using RGBA

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

## Background Image

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

/*background image set for specific elements*/
p { background-image: url("paper.gif"); }
```

## Background Repeat

By default, the background-image property repeats an image both horizontally and vertically.

### Horizontal only

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

### Vertical only

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

### No repeat

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

## Background Position

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

## Background Attachment (should scroll or be fixed)

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

## Shorthand property

Insead of

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

```
        background-position: right top;
    }
```

## Use the shorthand property background

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

# Borders

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

## Individual Sides

```
p {
    border-top-style: dotted;
    border-right-style: solid;
    border-bottom-style: dotted;
    border-left-style: solid;
}

/*or*/

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

## Shorthand Property

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

## Rounded Borders

```
p { border: 2px solid red; border-radius: 5px; }
```

# Margins

To create space around elements, outside of any defined borders.

```
p { margin-top: 100px; margin-bottom: 100px; margin-right: 150px; margin-left:
    80px; }
p { margin: 25px 50px 75px 100px; }
```

## auto Value

To horizontally center the element within its container

```
/*for the child element*/
```

```
.center {
  width:fit-content; /*without this the element may occupy the full length*/
  margin: auto;
  border: 1px solid red;
}
```

## inherit Value

Margin of child element will be inherited from the parent element

```
/*for the child element*/
.center {
  width:fit-content; /*without this the element may occupy the full length*/
  margin: inherit;
  border: 1px solid red;
}
```

## Margin Collapse (Happens automatically)

Top and bottom margins of elements are automatically collapsed into a single margin that is equal to the largest of the two margins. This does not happen on left and right margins! Only top and bottom margins!

```
h1 { margin: 0 0 50px 0; } h2 { margin: 20px 0 0 0; }
/*resultant margin will be 50px and not 70px*/
```

## Padding

To generate space around an element's content, inside of any defined borders.

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; }
div { padding: 25px 50px 75px 100px; }
```

**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; }
```

## Padding and Element Width

```
div { width: 300px; padding: 25px; }
/*Total width of the div will be 300+25+25=350px*/
```

# Height and Width

- auto - This is default. The browser calculates the height and width
- length - Defines the height/width in px, cm etc.
- % - Defines the height/width in percent of the containing block
- initial - Sets the height/width to its default value
- inherit - The height/width will be inherited from its parent value

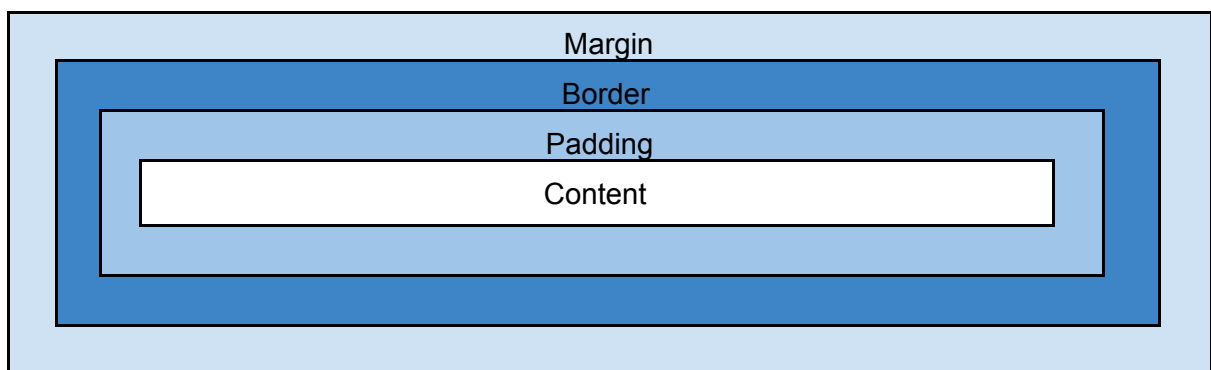
## Max-width

To set the maximum width of an element.

## Box Model

All HTML elements can be considered as boxes. The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.

```
div { width: 300px; border: 15px solid green; padding: 50px; margin: 20px; }
```



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

## Width and Height of an Element

**When you set the width and height properties of an element, you just set the width and height of the content area.**

## Outline

Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline.

None of the other outline properties will have ANY effect unless the outline-style property is set!

```
p.ex1 { border: 1px solid black; outline-style: solid; outline-color: red;
        outline-width: thin; }
p.ex1 {outline: dashed;}
p.ex2 {outline: dotted red;}
p.ex3 {outline: 5px solid yellow;}
p.ex4 {outline: thick ridge pink;}
```

## outline-style

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

## outline-width

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

## outline-offset

Adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

```
p { margin: 30px; border: 1px solid black; outline: 1px solid red;
    outline-offset: 15px; }
```

# Text

## Color

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

## Horizontal Alignment

To set the horizontal alignment of a text.

```
h1 { text-align: center; } h2 { text-align: left; } h3 { text-align: right; }  
div { text-align: justify; }
```

## Vertical Alignment

```
img.top { vertical-align: top; } img.middle { vertical-align: middle; }  
img.bottom { vertical-align: bottom; }
```

## Direction

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

## Underline, Overline, Line-through

```
a { text-decoration: none; } h1 { text-decoration: overline; }  
h2 { text-decoration: line-through; } h3 { text-decoration: underline; }
```

It is not recommended to underline text that is not a link, as this often confuses the reader.

## Uppercase and Lowercase

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

## Indentation

To specify the indentation of the first line of a text

```
p { text-indent: 50px; }
```

## Letter Spacing

To specify the space between the characters in a text

```
h1 { letter-spacing: 3px; } h2 { letter-spacing: -3px; }
```

## Space Between Lines

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

## Word Spacing

```
h1 { word-spacing: 10px; } h2 { word-spacing: -5px; }
```

## Wrapping

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

```
p { white-space: nowrap; }
```

## Text Shadow

```
/*Horizontal shadow, vertical shadow, blur effect, color*/  
h1 { text-shadow: 2px 2px 5px red; }
```

## Fonts

Font properties define the font family, boldness, size, and the style of a text.

### Font Families

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

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

```
.serif { font-family: "Times New Roman", Times, serif; }  
.sansserif { font-family: Arial, Helvetica, sans-serif; }  
.monospace { font-family: "Lucida Console", Courier, monospace; }
```

### Font Style

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

## Font Weight

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

## small-caps font

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

## Font Size

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.

## Pixels

Gives full control over the text size

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

## Em

1em = current font size. The default text size in browsers is 16px. pixels/16=em

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

## 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; }
```

## Responsive Font Size

```
<h1 style="font-size:10vw">Hello World</h1>
```

## Shorthand

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

# Icons

## Font Awesome Icons

Go to [fontawesome.com](https://fontawesome.com), sign in, and get a code.

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

```
<body>
    <i class="fas fa-cloud"></i>
</body>
```

## Bootstrap Icons

```
<head>
    <link rel="stylesheet"
          href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.mi
          n.css">
</head>
<body>
    <i class="glyphicon glyphicon-cloud"></i>
</body>
```

## Google Icons

```
<head>
    <link rel="stylesheet"
          href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>
    <i class="material-icons">cloud</i>
</body>
```

## Links

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

```
/* 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; }
```

## Text Decoration

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

```
a:link { background-color: yellow; }
a:visited { background-color: cyan; }
a:hover { background-color: lightgreen; }
a:active { background-color: hotpink; }
```

## Link 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; }
```

## How to apply style

```
<style>
    a.one:hover { ... }
</style>

<body>
    <a class="one" ...>...</a>
</body>
```

## Lists

### List Item Markers (Bullets)

```
ul.a { list-style-type: circle; } ul.b { list-style-type: square; }
ol.c { list-style-type: upper-roman; } ol.d { list-style-type: lower-alpha; }
```

### Image as The List Item Marker (Image Bullets)

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

### Position The List Item Markers

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

### Remove Default Settings

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

### Shorthand property

```
ul { list-style: square inside url("sqpurple.gif"); }
```

# Tables

## Borders

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

## Full-Width Table

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

## Remove Double Borders

double borders are because both the table and the <th> and <td> elements have separate borders.

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

To have single border add: `table { border-collapse: collapse; }`

## Border around the table only

```
table { border: 1px solid black; }
```

## Table Width and Height

```
table { width: 100%; } th { height: 70px; }
```

## Half-page table

```
table { width: 50%; } th { height: 70px; }
```

## Table Alignment

### Horizontal Alignment

```
th { text-align: left; }
```

### Vertical Alignment

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

## Table Style

### Table Padding

```
th, td { padding: 15px; text-align: left; }
```

## Horizontal Dividers

```
th, td { border-bottom: 1px solid #ddd; }
```

## Hoverable Table

```
tr:hover { background-color: #f5f5f5; }
```

## Striped Tables

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

## Table Color

```
th { background-color: #4CAF50; color: white; }
```

## Responsive Table (Scrollable)

Add a container element (like <div>) with overflow-x:auto around the <table> element to make it responsive:

```
<div style="overflow-x:auto;">
  <table>
    ... table content ...
  </table>
</div>
```

# Display (Layout Control)

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).

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.

Examples of inline elements:

<span>, <a>, <img>

## Hide and show elements

```
display: none; /*or*/ visibility: hidden;
```

## Override The Default Display Value (eg. for horizontal menu)

Changing an inline element to a block element, or vice versa.

```
/*Making inline <li> elements for horizontal menus*/  
li { display: inline; }
```

## display: inline vs display: inline-block

display: inline-block allows:

- To set a width and height on the element
- The top and bottom margins/paddings are respected

```
display: inline; height: 2em;"
```

```
✗ /*height doesn't work*/
```

```
display: inline-block; width: 100px; height: 50px; overflow: scroll; etc.."
```

```
✓ /*can set height and width*/
```

## position

### static

- HTML elements are positioned static by default
- Not affected by the top, bottom, left, and right properties.
- Always positioned according to the normal flow of the page

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

### relative

- Positioned relative to its normal position
- Will cause it to be adjusted away from its normal position by setting top, right, bottom, and left properties

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

### fixed

- Positioned relative to the viewport
- Always stays in the same place even if the page is scrolled
- The top, right, bottom, and left properties are used to position the element.

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

## absolute

- positioned relative to the nearest positioned ancestor
- NOT positioned relative to the viewport, like fixed
- A "positioned" element is one whose position is anything except static.

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

## sticky

- Positioned based on the user's scroll position
- Toggles between relative and fixed, depending on the scroll position
- Positioned relative until a given offset position is met in the viewport - then it "sticks" in place

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

## Overlapping Elements

- 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 +ve or -ve stack order

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

## Positioning Text In an Image

```
.container { position: relative; }
.topright { position: absolute; top: 8px; right: 16px; font-size: 18px; }
img { width: 100%; height: auto; opacity: 0.3; }

<div class="container">
  
  <div class="topright">Top Right</div>
</div>
```

## Example (diagram)

```
<div style="position: relative;">
  <div style="display: inline-block; width: 50px; height: 50px;
background-color: gainsboro"></div>

  /*Red div*/
  <div style="display: inline-block; width: 50px; height: 50px;
background-color: red; --INSERT FOLLOWING CODE HERE--"></div>

  <div style="display: inline-block; width: 50px; height: 50px;
background-color: gainsboro;"></div>
</div>
```

## static

position: **static**; top: 10px; left: 10px; /\*top and left has no effect\*/



## relative

position: **relative**; top: 10px; left: 10px;



## absolute

position: **absolute**; top: 10px; left: 10px;



# Overflow

- 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 only works for block elements with a specified height.**

## visible

Default. The overflow is not clipped. The content renders outside the element's box

```
<div style="width: 300px; height: 100px; overflow: visible;">Lorem ipsum ..</div>
```

## hidden

The overflow is clipped, and the rest of the content will be invisible

```
<div style="width: 300px; height: 100px; overflow: hidden;">Lorem ipsum ..</div>
```

## scroll

The overflow is clipped, and a scrollbar is added to see the rest of the content

```
<div style="width: 300px; height: 100px; overflow: scroll;">Lorem ipsum ..</div>
```

## auto

Similar to scroll, but it adds scrollbars only when necessary

```
<div style="width: 300px; height: 100px; overflow: auto;">Lorem ipsum ..</div>
```

## overflow-x and overflow-y

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

## Float

- For positioning and formatting content e.g. let an image float left to the text in a container
- Can be used to wrap text around images.
- Is responsive (can make responsive photo gallery with divs float:left)

```
img { float: right; }
```

## Clear

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

can have one of the following values:

- none - Allows floating elements on both sides. This is default
- left - No floating elements allowed on the left side
- right - No floating elements allowed on the right side
- both - No floating elements allowed on either the left or the right side
- inherit - The element inherits the clear value of its parent

## Menu (Navigation Links) with inline-block

```
li{ list-style-type: none; margin: 0; padding: 0; }  
li{ display: inline-block; padding: 10px; }  
/*inline-block can have width and height*/
```

```
<ul>  
  <li>One</li>  
  <li>Two</li>  
  <li>Three</li>  
  <li>Four</li>  
  <li>Lorem, ipsum dolor sit amet consectetur adipisicing elit.</li>  
</ul>  
/*The menu is responsive*/
```

# Align

## Vertical and Horizontal Center

### Easy way

```
.parent { display: flex; align-items: center; justify-content: center; }  
--or--  
.parent { display: flex; } .child { margin: auto; }
```

## Center Align Elements

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

```
<div style="margin: auto; width: 300px;">
```

## Center Align Text

```
text-align: center;
```

## Center an Image

```
img { display: block; margin-left: auto; margin-right: auto; width: 40%; }
```

## Left and Right Align - Using position

```
.right { position: absolute; right: 0px; width: 300px; border: 3px solid #73AD21;  
padding: 10px; }
```

## Left and Right Align - Using float

```
.right { float: right; width: 300px; border: 3px solid #73AD21; padding: 10px; }
```

## Center Vertically - Using padding

```
.center { padding: 70px 0; border: 3px solid green; }
```

## Center Vertically - Using Flexbox

```
.center { display: flex; justify-content: center; padding: 1em; }
```

## Opacity

### Transparent Image

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

## Transparent Hover Effect

```
img { opacity: 0.5; } img:hover { opacity: 1.0; }

/*Reversed hover effect*/
img:hover { opacity: 0.5; }
```

## Navigation Bar (Menu)

Navigation Bar = List of Links

### Vertical Navigation Bar

```
ul {
    margin: 0; /*removes top, bottom default margin and bullets*/
    padding: 0; /*removes default padding*/
    width: 200px; /*default is 100%*/
    background-color: #f1f1f1;

    /*for full height fixed*/
    height: 100%; position: fixed;
}
li a{
    display: block; /*removes default inline. Now width is 100%*/
    text-decoration: none; /*removes default underline*/
    padding: 8px 16px; text-align: center; color: black;
}

/*highlight active page or link*/
li a.active{ background-color: steelblue; color: white; }

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

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

### Horizontal Navigation Bar

```
ul{
    list-style-type: none; /*reoves bullets*/
    margin: 0;
    padding: 0; /*removes default padding*/
    overflow: hidden;
    background-color: #333;
}
li{ float: left; /*arranges items in horizontal order*/ }
li a{
    display: block;
    padding: 14px 16px;
    text-decoration: none;
```

```

        text-align: center;
        color: white;
    }
    .active{
        background-color: teal;
    }
    li a:hover:not(.active){
        background-color: #111;
    }
    .rightAlign{
        float: right;
    }
<ul>
    <li><a class="active" href="default.asp">Home</a></li>
    <li><a href="news.asp">News</a></li>
    <li><a href="contact.asp">Contact</a></li>
    <li class="rightAlign"><a href="about.asp">About</a></li>
</ul>

```

## Fixed Top

```
ul { position: fixed; top: 0; width: 100%; }
```

## Fixed Bottom

```
ul { position: fixed; bottom: 0; width: 100%; }
```

## Sticky Navbar

[https://www.w3schools.com/css/css\\_navbar\\_horizontal.asp](https://www.w3schools.com/css/css_navbar_horizontal.asp)

```

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

```

# Dropdown Menu

(Tested)

```

.dropdown{ display: inline-block; position: relative; }
a{ background-color: ghostwhite; text-decoration: none; color: black;
    display: block; padding: 10px; }
a:hover{ background-color: gray; color: white; }
.dropbtn{ width: 200px; padding: 10px; border: none;
    background-color: ghostwhite; }
.dropdown-content{ width: 200px; display: none; position: absolute;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2); }

/*show child when hover parent*/
.dropdown:hover .dropdown-content{ display: block; }

<div class="dropdown">
    <button class="dropbtn">Show</button>
    <div class="dropdown-content">
        <a href="#">One</a>

```

```

        <a href="#">Two</a>
        <a href="#">One Hundred</a>
    </div>
</div>

```

## Forms

```

div{ border-radius: 5px; background-color: ghostwhite; padding: 20px; width: 50%; }

input[type=text], select{ box-sizing: border-box; padding: 12px 20px;
    margin: 8px 0; /*top bottom only*/
    border: 1px solid #ccc; border-radius: 4px;

    /*display and width vertical-align the controls*/
    display: inline-block; width: 100%; }

input[type=submit]{ width: 100%; background-color: steelblue; color: white;
    padding: 14px 20px; margin: 8px 0; border: none; border-radius: 4px;
    cursor: pointer; }

input[type=submit]:hover{ background-color: skyblue; }

form{ margin-bottom: 0; /*form has a default bottom margin?*/ }

<div>
    <form action="">
        <label for="fname">First Name</label>
        <input type="text" id="fname" name="firstname">

        <label for="lname">Last Name</label>
        <input type="text" id="lname" name="lastname">

        <label for="country">Country</label>
        <select name="country" id="country">
            <option value="usa">USA</option>
            <option value="uk">UK</option>
            <option value="australia">Australia</option>
        </select>

        <input type="submit" value="Submit">
    </form>
</div>

```

## Units

### 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)

## Relative Lengths

Relative length units specify a length relative to another length property

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

## Specificity

For two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and applies it.

# Specificity Hierarchy

Inline styles > IDs > Classes, attributes and pseudo-classes > Elements and pseudo-elements

## Specificity Rules

### Equal specificity: the latest rule counts

```
h1 {background-color: yellow;}  
h1 {background-color: red;} ✓
```

### ID selectors have a higher specificity than attribute selectors

```
div#a {background-color: green;} ✓  
#a {background-color: yellow;}  
div[id=a] {background-color: blue;}
```

### Contextual selectors are more specific than a single element selector

From external CSS file:

```
#content h1 {background-color: red;}
```

In HTML file: ✓

```
<style>  
#content h1 {  
    background-color: yellow;  
}  
</style>
```

### A class selector beats any number of element selectors

```
.intro {background-color: yellow;} ✓  
h1 {background-color: red;}
```

# Backgrounds

## Multiple Backgrounds

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

## Fixed

If fixed image doesn't move on scroll

```
background: url(images/beach.webp) no-repeat center fixed;
```

## Background Size

```
background-size: 100px 80px;  
background-size: 50px, 130px, auto;
```

```
/*scales the background image to be as large as possible (but both its width and  
its height must fit inside the content area)*/
```

```
background-size: contain;
```

```
/*scales the background image so that the content area is completely covered by the  
background image*/
```

```
background-size: cover;
```

## Layout Modules

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

## Flexbox

A flex container with three flex items:

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

## Container (Parent Element)

```
.flex-container { display: flex; }
```

### Flex container properties

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

### flex-direction (Horizontal/Vertical)

```
flex-direction: /*column, column-reverse, row, row-reverse*/;
```

### flex-wrap

```
flex-wrap: /*wrap, nowrap, wrap-reverse,*/;
```

## flex-flow (Shorthand)

Shorthand property for setting both the flex-direction and flex-wrap

```
flex-flow: row wrap;
```

## justify-content (Main Axis)

Controls alignment of all items on the main axis.

```
justify-content: /*center, flex-start, flex-end, space-around, space-between*/;
```

## align-items (Cross Axis)

Controls alignment of all items on the cross axis.

```
align-items: /*center, flex-start, flex-end, stretch(default), baseline*/;
```

## baseline

The item with the largest distance between its cross-start margin edge and its baseline is flushed with the cross-start edge of the line.

## align-content

It helps to align a flex container's lines within it when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

```
align-content: /*space-between, space-around, stretch, center, flex-start, flex-end*/;
```

## Perfect Centering

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

## Flex Items (Child Elements)

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

## Flex Item Properties

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

## order

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

```
<div style="order: 4">3</div>
<div style="order: 1">4</div>
</div>
```

## flex-grow

Make the third flex item grow eight times faster than the other flex items

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```

## Flex-basis

The flex-basis property specifies the initial length of a flex item.

```
flex-basis: 200px
```

## flex

Shorthand property for the flex-grow, flex-shrink, and flex-basis

```
flex: 0 0 200px
```

## align-self

Aligns the item on the cross axis

```
align-self: /*center, flex-start, flex-end, etc*/;
```

## Responsive Flexbox

```
.flex-container {
  display: flex;
  flex-direction: row;
}

@media (max-width: 800px) {
  .flex-container {
    flex-direction: column;
  }
}
```

## Responsive Website using Flexbox

```
*{ font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  outline: 1px dotted gray; box-sizing: border-box; }
body{ margin: 0; }
.header{ padding: 60px; text-align: center; background-color: #1abc9c;
  color: white ; }
.navbar{ display: flex; background: #333; }
.navbar a{ padding: 14px 20px; color: white; text-decoration: none;
  text-align: center; }
```

```

.navbar a:hover{ background-color: #ddd; color: black; }
.side{ background-color: #f1f1f1; padding: 20px; }
.main{ background-color: white; padding: 20px; }
.fakeimg{ background-color: #aaa; padding: 20px; width: 100%; }
.footer{ text-align: center; padding: 20px; background-color: #ddd; }

/*Responsive - flex & media start*/
.row{ display: flex; flex-wrap: wrap; }
.side{ flex: 30%; }
.main{ flex: 70%; }
@media screen and (max-width: 700px) {
    .row, .navbar{ flex-direction: column; }
}
/*Responsive - flex & media end*/

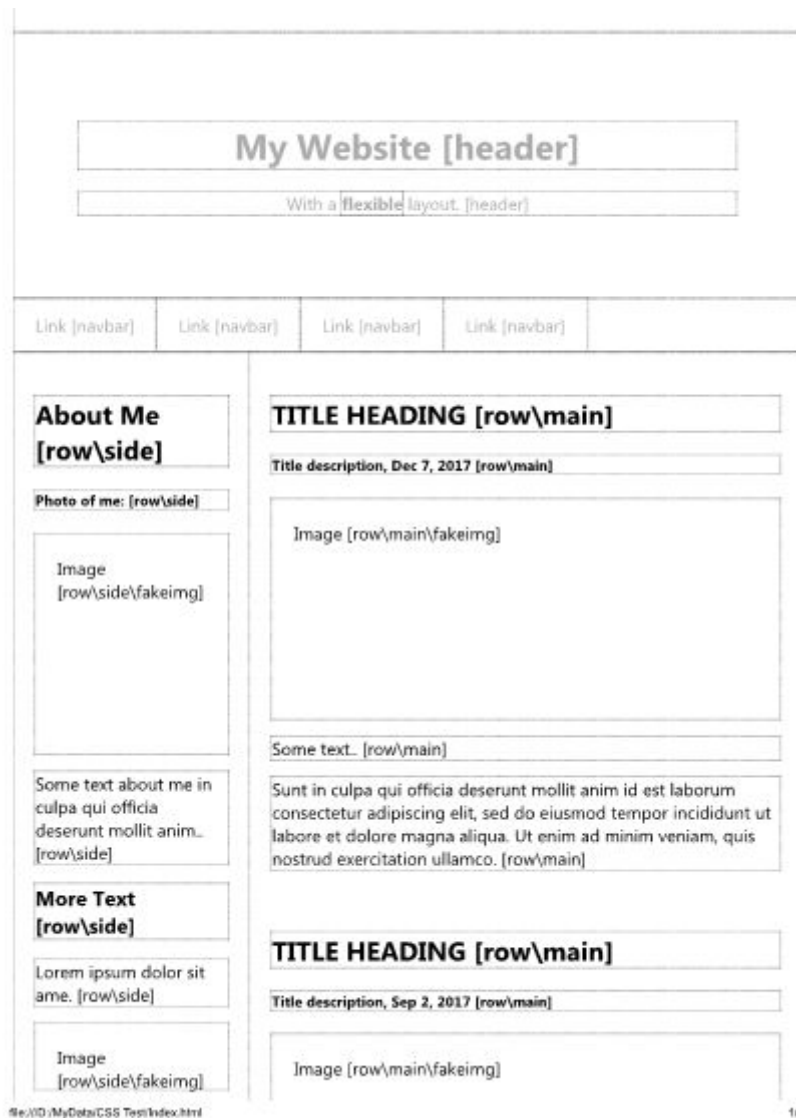
<div class="header">
    <h1>My Website</h1>
    <p>With a <b>flexible</b> layout.</p>
</div>

<div class="navbar">
    <a href="#">Link</a>
    <a href="#">Link</a>
    <a href="#">Link</a>
    <a href="#">Link</a>
</div>

<!-- The flexible grid (content) -->
<div class="row">
    <div class="side">
        <h2>About Me</h2>
        <h5>Photo of me</h5>
        <div class="fakeimg" style="height:200px;">Image</div>
        <p>Some text about me in culpa qui officia deserunt mollit anim..</p>
        <h3>More Text</h3>
        <p>Lorem ipsum dolor sit ame.</p>
        <div class="fakeimg" style="height:60px;">Image</div><br>
        <div class="fakeimg" style="height:60px;">Image</div><br>
        <div class="fakeimg" style="height:60px;">Image</div>
    </div>
    <div class="main">
        <h2>TITLE HEADING</h2>
        <h5>Title description, Dec 7, 2017</h5>
        <div class="fakeimg" style="height:200px;">Image</div>
        <p>Some text..</p>
        <p>Sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
        <br>
        <h2>TITLE HEADING</h2>
        <h5>Title description, Sep 2, 2017</h5>
        <div class="fakeimg" style="height:200px;">Image</div>
        <p>Some text.. </p>
        <p>Sunt in culpa qui officia deserunt mollit anim id est laborum..</p>
    </div>
</div>

<div class="footer">
    <h2>Footer [footer]</h2>
</div>

```



## Grid Layout Module

Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

### Grid Elements

- Grid layout consists of a parent element, with one or more child elements.
- A Grid Layout must have a parent element with the **display** property set to **grid** or **inline-grid**.
- Direct child element(s) of the grid container automatically becomes grid items.

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

## display

```
display: grid; or display: inline-grid;
```

## Grid Gaps

The spaces between each column/row are called gaps.

```
Grid-column-gap, grid-row-gap, grid-gap  
.grid-container { display: grid; grid-column-gap: 50px; }
```

## grid-gap property (shorthand)

```
grid-gap: 50px 100px; or grid-gap: 50px;
```

## Grid Lines

The lines between columns are called column lines. The lines between rows are called row lines. Starting from 1

Place a grid item at column line 1, and let it end on column line 3:

```
.item1 { grid-column-start: 1; grid-column-end: 3; }
```

## Grid Container

### No of columns + Width of columns

```
grid-template-columns: auto auto auto auto; /*4 columns. "auto" if all columns  
should have the same width*/
```

```
grid-template-columns: 80px 200px auto 40px;
```

### Height of each row

```
grid-template-rows: 80px 200px;
```

## Horizontal Alignment

```
justify-content: /*space-around, space-between, center, start, end*/
```

## Vertical Alignment

The grid's total height has to be less than the container's height for the align-content property to have any effect.

```
align-content: /*center, space-evenly, space-around, space-between, start, end*/
```

## Grid Item (Child Elements)

### Column Placement of an Item

To define where the item will start, and where the item will end. shorthand property for the `grid-column-start` and the `grid-column-end` properties.

```
grid-column: 1 / 5 /*start on column 1 and end before column 5*/  
grid-column: 1 / span 3; /*start on column 1 and span 3 columns*/
```

### Row Placement of an Item

`grid-row` property is a shorthand property for the `grid-row-start` and the `grid-row-end` properties

```
grid-row: 1 / 4; grid-row: 1 / span 2; (See grid-column)
```

### Row, Column Placement Shortcut

`grid-area` property can be used as a shorthand property for the `grid-row-start`, `grid-column-start`, `grid-row-end` and the `grid-column-end` properties

```
/*start on row-line 1 and column-line 2, and end on row-line 5 and column line 6*/  
grid-area: 1 / 2 / 5 / 6;
```

## Naming Grid Items

Item1 gets the name "myArea" and spans all five columns in a five columns grid layout

```
.item1 { grid-area: myArea; }  
.grid-container { grid-template-areas: 'myArea myArea myArea myArea myArea'; }
```

Period sign represents a grid item with no name

```
.grid-container { grid-template-areas: 'myArea myArea . . .'; }
```

Make "item1" span two columns and two rows

```
.grid-container { grid-template-areas: 'myArea myArea . . .' 'myArea myArea . . .';  
}
```

Name all items, and make a ready-to-use webpage template

```
.item1 { grid-area: header; }  
.item2 { grid-area: menu; }  
.item3 { grid-area: main; }  
.item4 { grid-area: right; }  
.item5 { grid-area: footer; }  
  
.grid-container {  
  grid-template-areas:  
    'header header header header header header'  
    'menu main main main right right'  
    'menu footer footer footer footer footer';  
}
```

## Order of the Items

```
.item1 { grid-area: 1 / 3 / 2 / 4; } .item2 { grid-area: 2 / 3 / 3 / 4; }  
.item3 { grid-area: 1 / 1 / 2 / 2; } .item4 { grid-area: 1 / 2 / 2 / 3; }  
.item5 { grid-area: 2 / 1 / 3 / 2; } .item6 { grid-area: 2 / 2 / 3 / 3; }
```

## Change Order Responsively

```
@media only screen and (max-width: 500px) {  
  .item1 { grid-area: 1 / span 3 / 2 / 4; }  
  .item2 { grid-area: 3 / 3 / 4 / 4; }  
  .item3 { grid-area: 2 / 1 / 3 / 2; }  
  .item4 { grid-area: 2 / 2 / span 2 / 3; }  
  .item5 { grid-area: 3 / 1 / 4 / 2; }  
  .item6 { grid-area: 2 / 3 / 3 / 4; }  
}
```

## Responsive Grid-View

[https://www.w3schools.com/css/css\\_rwd\\_grid.asp](https://www.w3schools.com/css/css_rwd_grid.asp)

First ensure that all HTML elements have the box-sizing property set to border-box. A responsive grid-view often has 12 columns, and has a total width of 100%. percentage for one column:  $100\% / 12 \text{ columns} = 8.33\%$

```
.col-1 {width: 8.33%;} .col-2 {width: 16.66%;} .col-3 {width: 25%;}  
.col-4 {width: 33.33%;} .col-5 {width: 41.66%;} .col-6 {width: 50%;}  
.col-7 {width: 58.33%;} .col-8 {width: 66.66%;} .col-9 {width: 75%;}  
.col-10 {width: 83.33%;} .col-11 {width: 91.66%;} .col-12 {width: 100%;}
```

```
[class*="col-"] {  
  float: left; /*All these columns should be floating to the left*/  
  padding: 15px; border: 1px solid red;  
}
```

```
<div class="row">  
  <div class="col-3">...</div> <!-- 25% →  
  <div class="col-9">...</div> <!-- 75% -->  
</div>
```

## Responsive Web Design - Media Queries

/\*When the screen (browser window) gets smaller than 768px, each column should have a width of 100%:\*/

**/\* For desktop: \*/**

```
.col-1 {width: 8.33%;} .col-2 {width: 16.66%;} .col-3 {width: 25%;}  
...  
.col-10 {width: 83.33%;} .col-11 {width: 91.66%;} .col-12 {width: 100%;}
```

**/\* For mobile phones: \*/**

```
@media only screen and (max-width: 768px) {  
  [class*="col-"] { width: 100%; }  
}
```

# Always Design for Mobile First

[https://www.w3schools.com/css/css\\_rwd\\_mediaqueries.asp](https://www.w3schools.com/css/css_rwd_mediaqueries.asp)

Instead of changing styles when the width gets smaller than 768px, we should change the design when the width gets larger than 768px. **This will make the page display faster on smaller devices**

```
/* For mobile phones (default) */
[class*="col-"] { width: 100%;}

/* For desktop: */
@media only screen and (min-width: 768px) {
  .col-1 {width: 8.33%;} .col-2 {width: 16.66%;} .col-3 {width: 25%;}
  ...
  .col-10 {width: 83.33%;} .col-11 {width: 91.66%;} .col-12 {width: 100%;}
}
```

## Another Breakpoint (Mobile, Tab, PC)

```
/* For mobile phones*/
[class*="col-"] {
  width: 100%;
}

/* For tablets*/
@media only screen and (min-width: 600px) {
  .col-s-1 {width: 8.33%;} .col-s-2 {width: 16.66%;} .col-s-3 {width: 25%;}
  ...
  .col-s-10 {width: 83.33%;} .col-s-11 {width: 91.66%;} .col-s-12 {width: 100%;}
}

/* For desktop*/
@media only screen and (min-width: 768px) {
  .col-1 {width: 8.33%;} .col-2 {width: 16.66%;} .col-3 {width: 25%;}
  ...
  .col-10 {width: 83.33%;} .col-11 {width: 91.66%;} .col-12 {width: 100%;}
}

<div class="row">
  <div class="col-3 col-s-3">...</div>
  <div class="col-6 col-s-9">...</div>
  <div class="col-3 col-s-12">...</div>
</div>
```

## Typical Device Breakpoints

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}
```

```
/* Extra large devices (large laptops and desktops, 1200px and up) */  
@media only screen and (min-width: 1200px) {...}
```

## Orientation: Portrait / Landscape

```
@media only screen and (orientation: landscape) {  
    body { background-color: lightblue; }  
}
```

## Hide Elements With Media Queries

```
@media only screen and (max-width: 600px) {  
    div.example { display: none; }  
}  
EoF
```