## Setting The Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

You should include the following <meta> viewport element in all your web pages:

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

A <meta> viewport element gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

CSS is a language that describes the style of an HTML document.

CSS describes how HTML elements should be displayed.

## What is CSS?

* **CSS** stands for **C**ascading **S**tyle **S**heets
* 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**

## CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



## CSS Example: tag /element Selector

body {  
    background-color: lightblue;  
}  
  
h1 {  
    color: white;  
    text-align: center;  
}  
  
p {  
    font-family: verdana;  
    font-size: 20px;  
}

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

# CSS How To...

## Three Ways to Insert CSS

There are three ways of inserting a style sheet:

* External style sheet
* Internal style sheet
* Inline style

## External Style Sheet

With an external style sheet, you can change the look of an entire website by changing just one file!

### Example

<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>

## Internal Style Sheet

An internal style sheet may be used if one single page has a unique style.

### Example

<head>  
<style>  
body {  
    background-color: linen;  
}  
  
h1 {  
    color: maroon;  
    margin-left: 40px;  
}   
</style>  
</head>

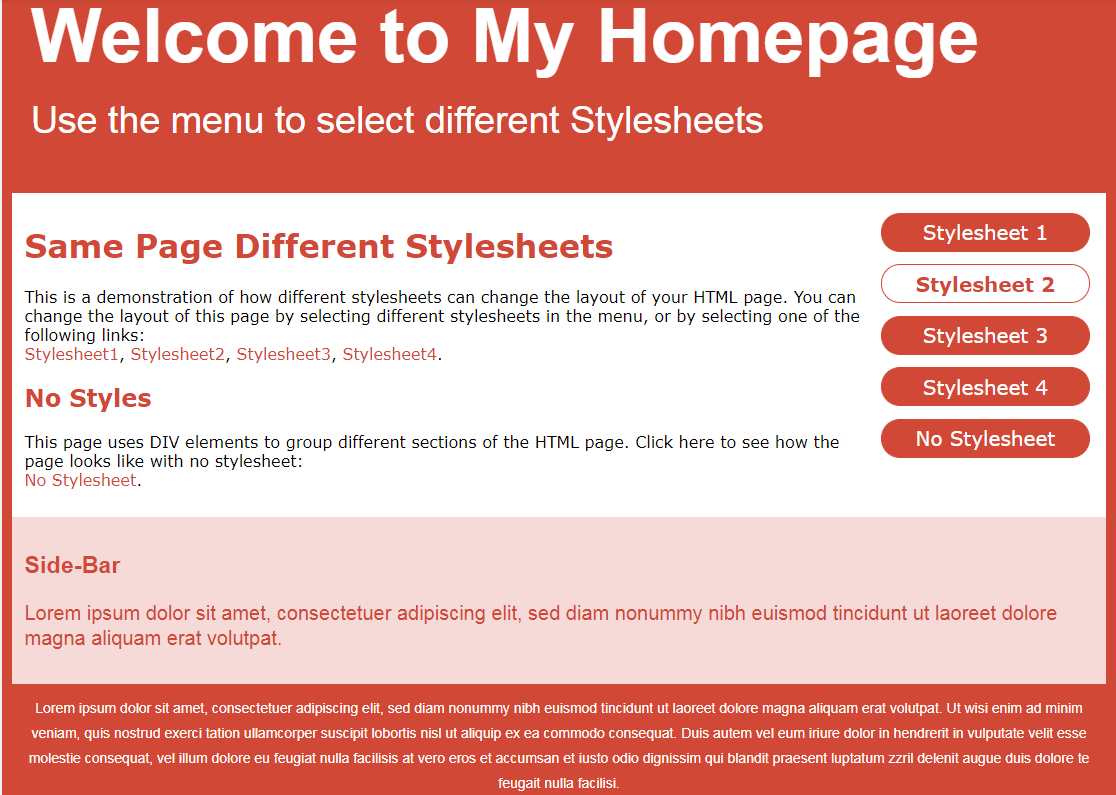
## Inline Styles

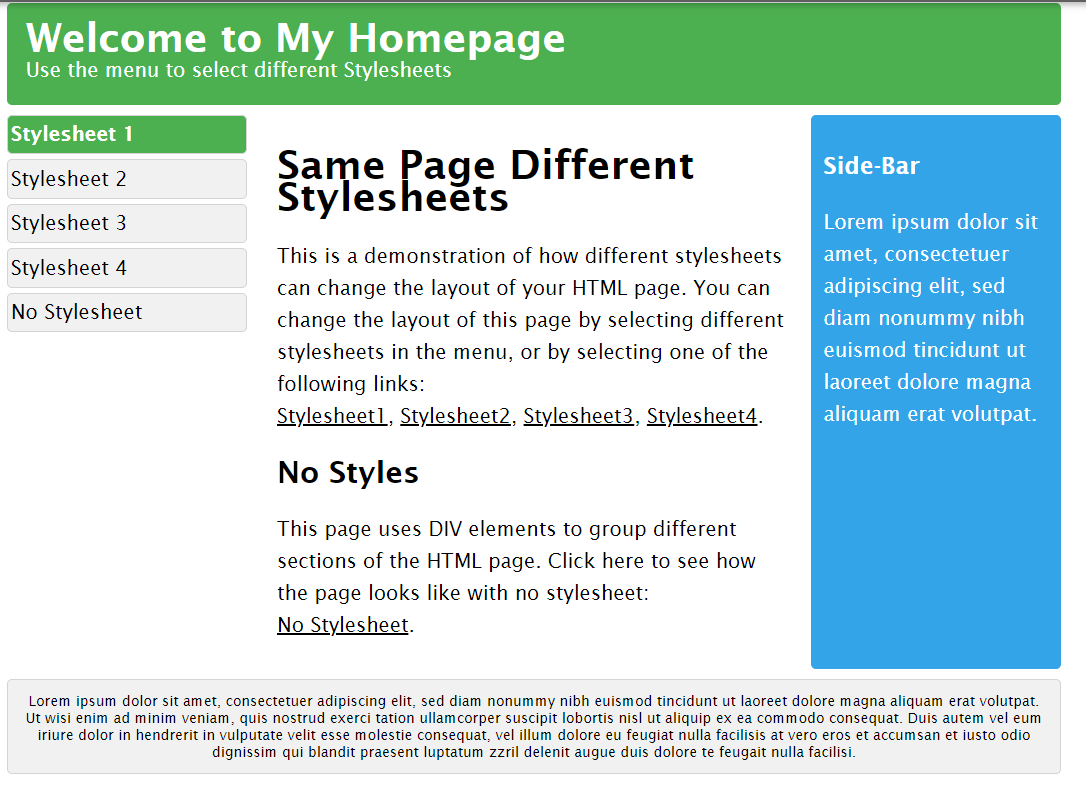
An inline style may be used to apply a unique style for a single element.

### Example

<h1 style="color:blue;margin-left:30px;">This is a heading</h1>

Template:





## Color Names

In HTML, a color can be specified by using a color name:



# CSS Backgrounds

The CSS background properties are used to define the background effects for elements.

CSS background properties:

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

## Background Color

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

### Example

body {  
    background-color: lightblue;  
}

## Background Image

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

### Example

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

body {  
    background-image: url("gradient\_bg.png");  
    background-repeat: repeat-x;  
}

body {  
    background-image: url("img\_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
}

body {  
    background-image: url("img\_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
    background-attachment: fixed;  
}

body {  
    background: #ffffff url("img\_tree.png") no-repeat right top;  
}

## CSS Border Properties

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

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

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;}

# CSS Margins

## CSS 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).

## Margin - 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

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

## CSS 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).

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

**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;  
}

# CSS Height and Width

## Setting height and width

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

The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in length values, like px, cm, etc., or in percent (%) of the containing block.

This element has a height of 200 pixels and a width of 50%

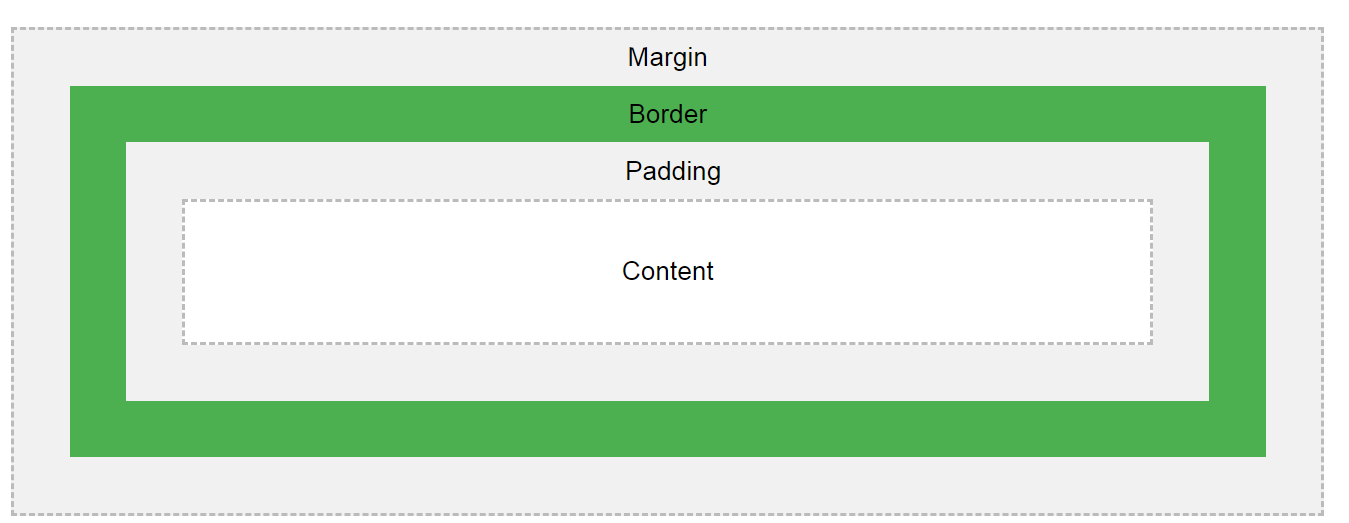
### Example

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

## The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

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



Explanation of the different parts:

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

The box model allows us to add a border around elements, and to define space between elements.

### Example

div {  
    width: 300px;  
    border: 25px solid green;  
    padding: 25px;  
    margin: 25px;  
}

# CSS Text

## 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):

### Example

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

# CSS Links

## Styling Links

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

### Example

a {  
    color: hotpink;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss_link_all)

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

### Example

/\* 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

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

### Example

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:

### Example

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

## Advanced - Link Buttons

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

### Example

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;  
}

# CSS Lists

## Unordered Lists:

* Coffee
* Tea
* Coca Cola
* Coffee
* Tea
* Coca Cola

## Ordered Lists:

1. Coffee
2. Tea
3. Coca Cola
4. Coffee
5. Tea
6. Coca Cola

## Different List Item Markers

The list-style-type property specifies the type of list item marker.

The following example shows some of the available list item markers:

### Example

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;  
}

# CSS Tables

## Vertical Alignment

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

By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements).

The following example sets the vertical text alignment to bottom for <td> elements:

### Example

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

## able Padding

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

### Example

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

## Striped Tables

|  |  |  |
| --- | --- | --- |
| **First Name** | **Last Name** | **Savings** |
| Peter | Griffin | $100 |
| Lois | Griffin | $150 |
| Joe | Swanson | $300 |

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

### Example

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

## 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>

# CSS Layout - 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

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;

Here is the CSS that is used:

### Example

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:

### Example

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:

### Example

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:** A "positioned" element is one whose position is anything except static.

Here is a simple example:

This <div> element has position: relative;

This <div> element has position: absolute;

Here is the CSS that is used:

### Example

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;  
}

# CSS Layout - Overflow

## overflow: visible

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

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

### Example

div {  
    width: 200px;  
    height: 50px;  
    background-color: #eee;  
    overflow: visible;  
}

# CSS Layout - float and clear

img {  
    float: right;  
}

# CSS Layout - inline-block

Example

.floating-box {  
    display: inline-block;  
    width: 150px;  
    height: 75px;  
    margin: 10px;  
    border: 3px solid #73AD21;   
}

## Demo: Dropdown Examples

Example

<style>  
/\* Style The Dropdown Button \*/  
.dropbtn {  
    background-color: #4CAF50;  
    color: white;  
    padding: 16px;  
    font-size: 16px;  
    border: none;  
    cursor: pointer;  
}  
  
/\* The container <div> - needed to position the dropdown content \*/  
.dropdown {  
    position: relative;  
    display: inline-block;  
}  
  
/\* Dropdown Content (Hidden by Default) \*/  
.dropdown-content {  
    display: none;  
    position: absolute;  
    background-color: #f9f9f9;  
    min-width: 160px;  
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);  
    z-index: 1;  
}  
  
/\* Links inside the dropdown \*/  
.dropdown-content a {  
    color: black;  
    padding: 12px 16px;  
    text-decoration: none;  
    display: block;  
}  
  
/\* Change color of dropdown links on hover \*/  
.dropdown-content a:hover {background-color: #f1f1f1}  
  
/\* Show the dropdown menu on hover \*/  
.dropdown:hover .dropdown-content {  
    display: block;  
}  
  
/\* Change the background color of the dropdown button when the dropdown content is shown \*/  
.dropdown:hover .dropbtn {  
    background-color: #3e8e41;  
}  
</style>  
  
<div class="dropdown">  
  <button class="dropbtn">Dropdown</button>  
  <div class="dropdown-content">  
    <a href="#">Link 1</a>  
    <a href="#">Link 2</a>  
    <a href="#">Link 3</a>  
  </div>  
</div>

# CSS Image Gallery

## Image Gallery

The following image gallery is created with CSS:

### Example

<html>  
<head>  
<style>  
div.gallery {  
    margin: 5px;  
    border: 1px solid #ccc;  
    float: left;  
    width: 180px;  
}  
  
div.gallery:hover {  
    border: 1px solid #777;  
}  
  
div.gallery img {  
    width: 100%;  
    height: auto;  
}  
  
div.desc {  
    padding: 15px;  
    text-align: center;  
}  
</style>  
</head>  
<body>  
  
<div class="gallery">  
  <a target="\_blank" href="fjords.jpg">  
    <img src="fjords.jpg" alt="Fjords" width="300" height="200">  
  </a>  
  <div class="desc">Add a description of the image here</div>  
</div>  
  
<div class="gallery">  
  <a target="\_blank" href="forest.jpg">  
    <img src="forest.jpg" alt="Forest" width="300" height="200">  
  </a>  
  <div class="desc">Add a description of the image here</div>  
</div>  
  
<div class="gallery">  
  <a target="\_blank" href="lights.jpg">  
    <img src="lights.jpg" alt="Northern Lights" width="300" height="200">  
  </a>  
  <div class="desc">Add a description of the image here</div>  
</div>  
  
<div class="gallery">  
  <a target="\_blank" href="mountains.jpg">  
    <img src="mountains.jpg" alt="Mountains" width="300" height="200">  
  </a>  
  <div class="desc">Add a description of the image here</div>  
</div>  
  
</body>  
</html>

# CSS Forms

### Example

input {  
    width: 100%;  
}

Example

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

Example

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

Example

input[type=text]:focus {  
    border: 3px solid #555;  
}

input[type=text] {  
    background-color: white;  
    background-image: url('searchicon.png');  
    background-position: 10px 10px;   
    background-repeat: no-repeat;  
    padding-left: 40px;  
}

input[type=text] {  
    -webkit-transition: width 0.4s ease-in-out;  
    transition: width 0.4s ease-in-out;  
}  
  
input[type=text]:focus {  
    width: 100%;  
}

Example

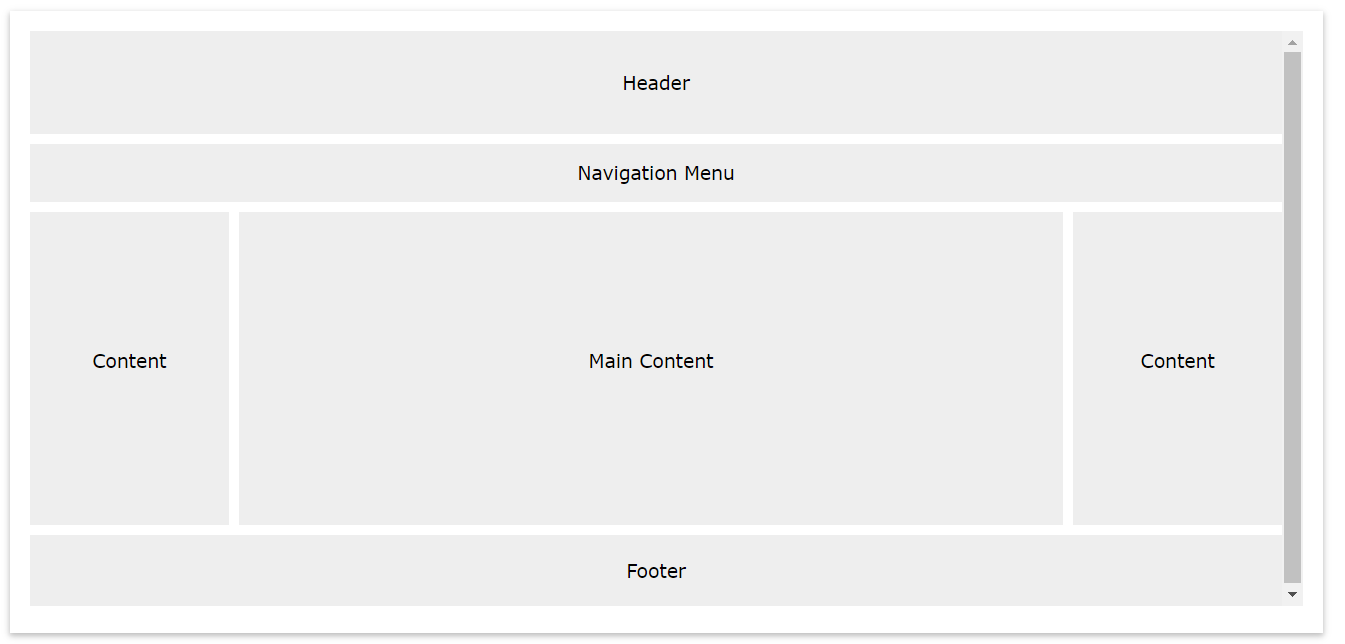
textarea {  
    width: 100%;  
    height: 150px;  
    padding: 12px 20px;  
    box-sizing: border-box;  
    border: 2px solid #ccc;

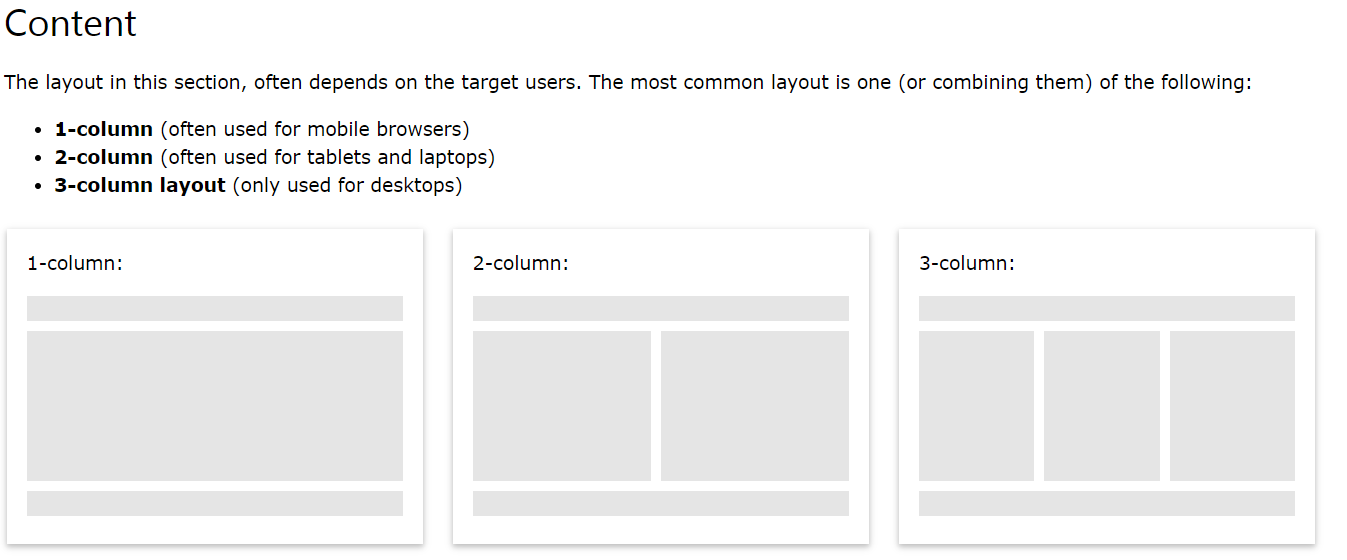
    border-radius: 4px;  
    background-color: #f8f8f8;  
    resize: none;  
}

Example

input[type=button], input[type=submit], input[type=reset] {  
    background-color: #4CAF50;  
    border: none;  
    color: white;  
    padding: 16px 32px;  
    text-decoration: none;  
    margin: 4px 2px;  
    cursor: pointer;  
}

# CSS Website Layout





## CSS border-radius Property

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

**Tip:** This property allows you to add rounded borders to elements!

Example

#rcorners1 {  
    border-radius: 25px;  
    background: #73AD21;  
    padding: 20px;   
    width: 200px;  
    height: 150px;   
}  
  
#rcorners2 {  
    border-radius: 25px;  
    border: 2px solid #73AD21;  
    padding: 20px;   
    width: 200px;  
    height: 150px;   
}  
  
#rcorners3 {  
    border-radius: 25px;  
    background: url(paper.gif);  
    background-position: left top;  
    background-repeat: repeat;  
    padding: 20px;   
    width: 200px;  
    height: 150px;   
}

Example

#borderimg1 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 50 round;  
}  
  
#borderimg2 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 20% round;  
}  
  
#borderimg3 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 30% round;  
}

# CSS Multiple Backgrounds

Example

#example1 {  
    background-image: url(img\_flwr.gif), url(paper.gif);  
    background-position: right bottom, left top;  
    background-repeat: no-repeat, repeat;  
}

Example

#example1 {  
    background: url(img\_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;  
}

## RGBA Colors

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

An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

rgba(255, 0, 0, 0.2);

rgba(255, 0, 0, 0.4);

rgba(255, 0, 0, 0.6);

rgba(255, 0, 0, 0.8);

Example

#p1 {background-color: rgba(255, 0, 0, 0.3);}  /\* red with opacity \*/  
#p2 {background-color: rgba(0, 255, 0, 0.3);}  /\* green with opacity \*/  
#p3 {background-color: rgba(0, 0, 255, 0.3);}  /\* blue with opacity \*/

**Linear Gradient - Top to Bottom (this is default)**

The following example shows a linear gradient that starts at the top. It starts red, transitioning to yellow:

Example

#grad {  
    background: linear-gradient(red, yellow);  
}