A CSS rule consists of a selector and a declaration block.

CSS Syntax



* Simple selectors (select elements based on name, id, class)
* [Combinator selectors](https://www.w3schools.com/css/css_combinators.asp) (select elements based on a specific relationship between them)
* [Pseudo-class selectors](https://www.w3schools.com/css/css_pseudo_classes.asp) (select elements based on a certain state)
* [Pseudo-elements selectors](https://www.w3schools.com/css/css_pseudo_elements.asp) (select and style a part of an element)
* [Attribute selectors](https://www.w3schools.com/css/css_attribute_selectors.asp) (select elements based on an attribute or attribute value)

There are three ways of inserting a style sheet:

* External CSS
* Internal CSS
* Inline CSS

<link rel="stylesheet" href="mystyle.css">

<h1 style="background-color:rgb(255, 99, 71);">...</h1>  
<h1 style="background-color:#ff6347;">...</h1>  
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>  
  
<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>  
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>

* background-color
* background-image
* background-repeat
* background-attachment
* background-position
* background (shorthand property)

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

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

**Tip:** To repeat an image vertically, set background-repeat: repeat-y;

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

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

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

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

Demonstration of the different border styles:

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;}

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

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

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

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

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

No border.

A hidden border.

A mixed border.

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:

The border-color property is used to set the color of the four borders.

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%)"
* transparent

If the border-style property has four values:

* **border-style: dotted solid double dashed;**
  + top border is dotted
  + right border is solid
  + bottom border is double
  + left border is dashed

If the border-style property has three values:

* **border-style: dotted solid double;**
  + top border is dotted
  + right and left borders are solid
  + bottom border is double

If the border-style property has two values:

* **border-style: dotted solid;**
  + top and bottom borders are dotted
  + right and left borders are solid

If the border-style property has one value:

* **border-style: dotted;**
  + all four borders are dotted

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

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

p {  
  border: 5px solid red;  
}

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

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).

Padding - 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

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

The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

CSS height and width Values

The height and width properties may have the following values:

* 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
* Setting max-width
* The max-width property is used to set the maximum width of an element.
* The max-width can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).
* The problem with the <div> above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.
* Using max-width instead, in this situation, will improve the browser's handling of small windows.
* **Tip:** Drag the browser window to smaller than 500px wide, to see the difference between the two divs!
* This element has a height of 100 pixels and a max-width of 500 pixels.
* **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).
* **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 total width of an element should be calculated like this:
* Total element width = width + left padding + right padding + left border + right border + left margin + right margin
* The total height of an element should be calculated like this:
* Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

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

CSS has the following outline properties:

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

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

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

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)

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

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

p.ex1 {outline: dashed;}  
p.ex2 {outline: dotted red;}  
p.ex3 {outline: 5px solid yellow;}  
p.ex4 {outline: thick ridge pink;}

* 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):

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):

p.a {  
  text-align-last: right;  
}  
  
p.b {  
  text-align-last: center;  
}  
  
p.c {  
  text-align-last: justify;  
}

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

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;  
}

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

h1 {  
  text-decoration-line: overline;  
  text-decoration-color: red;  
}  
  
h2 {  
  text-decoration-line: line-through;  
  text-decoration-color: blue;  
}  
  
h3 {  
  text-decoration-line: underline;  
  text-decoration-color: green;  
}  
  
p {  
  text-decoration-line: overline underline;  
  text-decoration-color: purple;  
}

h1 {  
  text-decoration-line: underline;  
  text-decoration-style: solid;  
}  
  
h2 {  
  text-decoration-line: underline;  
  text-decoration-style: double;  
}  
  
h3 {  
  text-decoration-line: underline;  
  text-decoration-style: dotted;  
}  
  
p.ex1 {  
  text-decoration-line: underline;  
  text-decoration-style: dashed;  
}  
  
p.ex2 {  
  text-decoration-line: underline;  
  text-decoration-style: wavy;  
}  
  
p.ex3 {  
  text-decoration-line: underline;  
  text-decoration-color: red;  
  text-decoration-style: wavy;  
}

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

}

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)

### **Example**

h1 {  
  text-decoration: underline;  
}  
  
h2 {  
  text-decoration: underline red;  
}  
  
h3 {  
  text-decoration: underline red double;  
}  
  
p {  
  text-decoration: underline red double 5px;  
}

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

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

### **Example**

p {  
  text-indent: 50px;  
}

h1 {  
  letter-spacing: 5px;  
}

p.small {  
  line-height: 0.8;  
}

p.one {  
  word-spacing: 10px;  
}

p {  
  white-space: nowrap;  
}

The text-shadow property adds shadow to text.

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

## Text shadow effect!

### **Example**

h1 {  
  text-shadow: 2px 2px;  
}

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

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.

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)

### **Example**

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

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

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

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

The font property is a shorthand property for:

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

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

### **Example**

Use font to set several font properties in one declaration:

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

* 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

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;  
}

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.

ADVERTISEMENT

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.

li {  
  display: inline;  
}

span {  
  display: block;  
}

h1.hidden {  
  display: none;  
}

h1.hidden {  
  visibility: hidden;  
}

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;

## 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.

## 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.

## 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.

## 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.

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

The 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).

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

**Note:** z-index only works on [positioned elements](https://www.w3schools.com/css/css_positioning.asp) (position: absolute, position: relative, position: fixed, or position: sticky) and [flex items](https://www.w3schools.com/css/css3_flexbox.asp) (elements that are direct children of display: flex elements).

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

**Note:** The overflow property only works for block elements with a specified height.

**Note:** In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

overflow: visible

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

## overflow: hidden

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

The CSS float property specifies how an element should float.

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

The float Property

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

The float property can have one of the following values:

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

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

The clear Property

When we use the float property, and we want the next element below (not on right or left), we will have to use the clear property.

The clear property specifies what should happen with the element that is next to a floating element.

The clear property can have one of the following values:

* none - The element is not pushed below left or right floated elements. This is default
* left - The element is pushed below left floated elements
* right - The element is pushed below right floated elements
* both - The element is pushed below both left and right floated elements
* inherit - The element inherits the clear value from its parent

When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

CSS Combinators

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

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 (~)

div p {  
  background-color: yellow;  
}

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

### **Example**

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:

### **Example**

div ~ p {  
  background-color: yellow;  
}

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
* Syntax
* The syntax of pseudo-classes:
* selector:pseudo-class {  
    property: value;  
  }

/\* 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;  
}

## 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:

### **Example**

p:first-child {  
  color: blue;  
}

<!DOCTYPE html>

<html>

<head>

<style>

p:first-child {

color: blue;

}

</style>

</head>

<body>

<p>This is some text.</p>

<p>This is some text.</p>

<div>

<p>This is some text.</p>

<p>This is some text.</p>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<style>

p i:first-child {

color: blue;

}

</style>

</head>

<body>

<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>

<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>

</body>

</html>

<style>

p:first-child i {

color: blue;

}

</style>

</head>

<body>

<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>

<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>

<div>

<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>

<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>

</div>

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

Syntax

The syntax of pseudo-elements:

selector::pseudo-element {  
  property: value;  
}

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

**Note:** The ::first-line pseudo-element can only be applied to block-level elements.

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

**Note:** The ::first-letter pseudo-element can only be applied to block-level elements.

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

## 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:

### **Example**

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:

### **Example**

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

## 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:

### **Example**

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

<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="img\_5terre.jpg">  
    <img src="img\_5terre.jpg" alt="Cinque Terre" width="600" height="400">  
  </a>  
  <div class="desc">Add a description of the image here</div>  
</div>  
  
<div class="gallery">  
  <a target="\_blank" href="img\_forest.jpg">  
    <img src="img\_forest.jpg" alt="Forest" width="600" height="400">  
  </a>  
  <div class="desc">Add a description of the image here</div>  
</div>  
  
<div class="gallery">  
  <a target="\_blank" href="img\_lights.jpg">  
    <img src="img\_lights.jpg" alt="Northern Lights" width="600" height="400">  
  </a>  
  <div class="desc">Add a description of the image here</div>  
</div>  
  
<div class="gallery">  
  <a target="\_blank" href="img\_mountains.jpg">  
    <img src="img\_mountains.jpg" alt="Mountains" width="600" height="400">  
  </a>  
  <div class="desc">Add a description of the image here</div>  
</div>  
  
</body>  
</html>

## 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:

### **Example**

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:

### **Example**

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":

### **Example**

[title~="flower"] {  
  border: 5px solid yellow;  
}

<!DOCTYPE html>

<html>

<head>

<style>

\* {

box-sizing: border-box;

}

input[type=text], select, textarea {

width: 100%;

padding: 12px;

border: 1px solid #ccc;

border-radius: 4px;

resize: vertical;

}

label {

padding: 12px 12px 12px 0;

display: inline-block;

}

input[type=submit] {

background-color: #04AA6D;

color: white;

padding: 12px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

float: right;

}

input[type=submit]:hover {

background-color: #45a049;

}

.container {

border-radius: 5px;

background-color: #f2f2f2;

padding: 20px;

}

.col-25 {

float: left;

width: 25%;

margin-top: 6px;

}

.col-75 {

float: left;

width: 75%;

margin-top: 6px;

}

/\* Clear floats after the columns \*/

.row::after {

content: "";

display: table;

clear: both;

}

/\* Responsive layout - when the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other \*/

@media screen and (max-width: 600px) {

.col-25, .col-75, input[type=submit] {

width: 100%;

margin-top: 0;

}

}

</style>

</head>

<body>

<h2>Responsive Form</h2>

<p>Resize the browser window to see the effect. When the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other.</p>

<div class="container">

<form action="/action\_page.php">

<div class="row">

<div class="col-25">

<label for="fname">First Name</label>

</div>

<div class="col-75">

<input type="text" id="fname" name="firstname" placeholder="Your name..">

</div>

</div>

<div class="row">

<div class="col-25">

<label for="lname">Last Name</label>

</div>

<div class="col-75">

<input type="text" id="lname" name="lastname" placeholder="Your last name..">

</div>

</div>

<div class="row">

<div class="col-25">

<label for="country">Country</label>

</div>

<div class="col-75">

<select id="country" name="country">

<option value="australia">Australia</option>

<option value="canada">Canada</option>

<option value="usa">USA</option>

</select>

</div>

</div>

<div class="row">

<div class="col-25">

<label for="subject">Subject</label>

</div>

<div class="col-75">

<textarea id="subject" name="subject" placeholder="Write something.." style="height:200px"></textarea>

</div>

</div>

<br>

<div class="row">

<input type="submit" value="Submit">

</div>

</form>

</div>

</body>

</html>

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):

/\* 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) {...}