

CSS

Pau Fernández

Universitat Politècnica de Catalunya (UPC)

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

CSS

CSS (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language, such as HTML.

- Enables the **separation** of content and presentation.
- It is composed of a list of **rules**, describing the style of a set of elements.
- Rules use **selectors** to describe the sets of elements to which the rule applies.
- **Cascading**: specified priority scheme to determine which rule applies.

Separation between content and presentation

CSS Zen Garden

<http://www.csszengarden.com/>



A demonstration of what can be accomplished through CSS-based design. Select any style sheet from the list to load it into this page.

Download the example  HTML FILE and  CSS FILE

THE ROAD TO ENLIGHTENMENT

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, broken CSS support, and abandoned browsers.

MID CENTURY
MODERN

by Andrew Lohman

GARMENTS

by Dan Mall

STEEL

by Steffen Knoeller

Block-level and inline-level elements

The page is a sequence of **block** elements, and a block element is made of **inline** elements.

Block

Represents a horizontal rectangle in a page.

Inline

Represent a rectangle in the line of elements inside a block.

Block-level elements (1)

Block elements generate a separate **box**.

They produce breaks before and after the element box. *They can't have other elements at the sides.*

Most structural elements are block: `header`, `footer`, `h1` to `h6`, `p`, `section`, `pre`, `ul`, `ol`, `li`, etc.

Generic element: `div`.

Block-level elements (2)

Content before

Block-level element

`<aside>`

Content after

Inline-level elements (1)

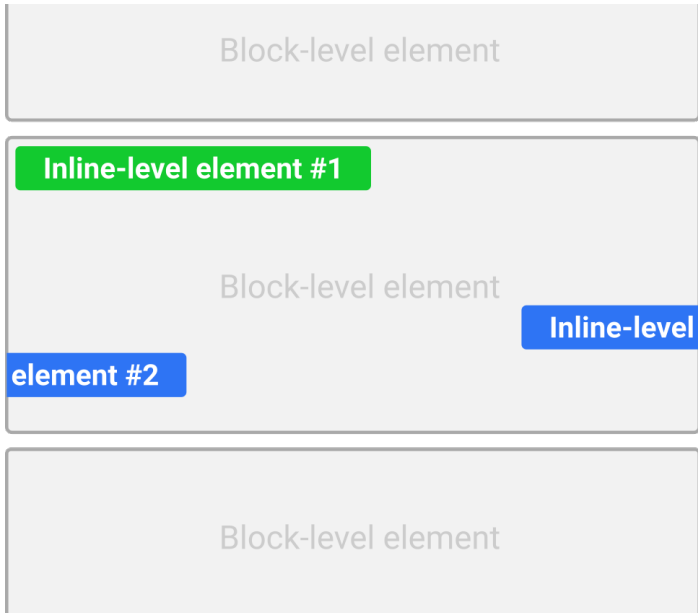
Inline elements distribute themselves along the current line (inside the current block element).

They don't break the flow of the line. They don't start a new line and only take up as much space as necessary.

All text-level elements are inline: `a` , `em` , `strong` , `input` , `button` , `img` , `code` , etc.

Generic element: `span` .

Inline-level elements



Styling HTML

External Stylesheet

A `link` in the `head` of the document sets the stylesheet.
index.html

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

style.css

```
body { font-family: sans-serif; }
...
```

Internal Stylesheet

Directly in the **head** of the document:

```
<html>
  <head>
    <style>
      body { font-family: sans-serif; }
      ...
    </style>
  </head>
  <body>
    ...
  </body>
</html>
```

@import a stylesheet

From within CSS, other files can be imported

index.html

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="main.css" />
    ...
  </head>
  ...
</html>
```

main.css

```
@import url(extra.css);
...
```

Inline styles

Styles can be specified directly on elements with the global `style` attribute.

```
<p style="color: gray;">
```

This paragraph will be printed in

```
<span style="background-color: yellow">gray</span> ,
```

except for one word, which has a different background.

```
</p>
```

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

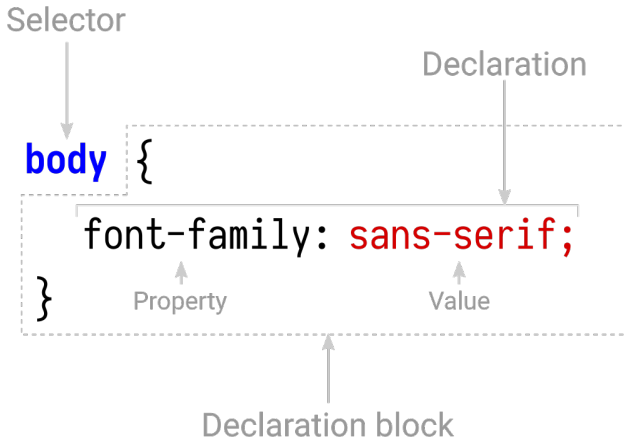
CSS Libraries & Tools

Syntax

A CSS stylesheet is just a **list of rules**

Each rule is: **selector**, a **property** and a **value**.

CSS Rules



Joining rules

Separate rules

```
h1 {  
  font-size: 20px;  
}  
h1 {  
  color: blue;  
}
```

can be joined together

```
h1 {  
  font-size: 20px;  
  color: blue;  
}
```

But every rule is *independent of every other*.

Whitespace doesn't matter

This CSS

```
h1 {  
  font-size: 20px;  
  color: blue;  
}  
p {  
  margin: 0;  
}
```

is equivalent to

```
h1 { font-size: 20px; color: blue; }  
p { margin: 0; }
```

and to

```
h1{font-size:20px;color:blue;}p{margin:0;}
```

Comments

Comments start with `/*` and end with `*/`.

```
/* Single line comment */  
p {  
  background-color: black;  
  color: white; /* Another single line comment */  
}  
strong {  
  /* This is a  
    multiline  
    comment */  
  color: red;  
}
```

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

Selectors

Selector specify a set

Every selector in a rule specifies a **set of elements** to whom we want to apply the rule.

Examples:

- All elements of type `h1`.
- All elements of class `important`.
- The element with ID `content`.
- All rows of a table inside the footer.
- All `span` elements having class `person` inside an `li` within an `ol` on the `article`.

Element type selector

tag : set of all elements of type **tag** .

All first heading elements are red

```
h1 { color: red; }
```

All paragraphs have 20 pixels of margin to the left

```
p { margin-left: 20px; }
```

The footer has a grey background

```
footer { background-color: #ddd; }
```

ID selectors

#id : specific element with ID id.

```
<html>
  <body>
    <p id="abstract">Lorem ipsum...</p>
    <p>dolor sit amet</p>
  </body>
</html>
```

We style the paragraph with ID `abstract` with

```
#abstract {
  border: 1px solid blue;
}
```

Class selector

.class : all elements having class `class`.

```
<body>
  <p>Lorem ipsum...</p>
  <p class="special">dolor sit amet</p>
  <p>consectetur adipiscing elit</p>
  <p>sed <span class="special">do eiusmod</span> tempor</p>
</body>
```

We style the set of elements with class `special` with

```
.special {
  background-color: orange;
}
```

Selector combinations

Comma combination

sel1, sel2 : All elements selected by **sel1** or **sel2** .

```
p, pre {  
  font-size: 18px;  
}  
ul, ol, .list {  
  margin-top: 1em;  
}  
#user-avatar, #logo, #search-bar {  
  padding: 0;  
}
```

Grouping

The comma enables **rule grouping**.

Rules sharing the same properties and values

```
h1 { color: red; }  
h2 { color: red; }  
h3 { color: red; }
```

can be joined together

```
h1, h2, h3 { color: red; }
```

Descendant selector

sel1 sel2 : all elements **sel2** inside a **sel1**

```
<div>
  <p>Hey Bee <span>See</span></p>
  <ul>
    <li><span>A</span></li>
    <li>B</li>
  </ul>
</div>
```

We style the set of elements with class **special** with

```
div span {
  font-weight: 600;
}
```


Descendant examples

An `em` inside a `h1` or `h2`

```
h1 em, h2 em {  
  text-decoration: underline;  
}
```

An element of class `icon` inside an `a` in a `li`

```
li a .icon {  
  margin: .5em;  
}
```

The paragraphs inside the element with id `main`

```
#main p {  
  font-family: serif;  
}
```

Child selector

tag1 > tag2 : all **tag2** which are direct children of **tag1** .

```
<body>
  <p>First <span>paragraph</span></p>
  <p>Second paragraph</p>
  <p><a href="/third">Third <span>paragraph</span></a></p>
</body>
```

We can target only the first span with

```
p > span {
  color: blue;
}
```

Tag and class

tag.class : all **tag**s which have a certain class.

```
<body>
  <p>Lorem ipsum...</p>
  <p class="special">dolor sit amet</p>
  <p>consectetur adipiscing elit</p>
  <p>sed <span class="special">do eiusmod</span> tempor</p>
</body>
```

We style the **span** with class **special** with

```
span.special {
  background-color: orange;
}
```

Attribute selector

tag[attr] : selects a **tag** with an attribute **attr** .

```
<body>
  <h1>Heading 1</h1>
  <h1 class="special">Special heading</h1>
  <h1 class="other">Other Heading</h1>
</body>
```

We can style **h1**s that *have* a class (independently of which one)

```
h1[class] {
  color: gray;
}
```

Attribute and value selector

tag[attr=value] : selects a **tag** with **attr** equal to **value** .

```
<nav>
  <a href="/home">Home</a>
  <a href="/blog">Blog</a>
  <a href="/about">About</a>
</nav>
```

We can style **a** elements which link to **"/home"**.

```
a[href="/home"] {
  color: pink;
}
```

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

Values & Properties

Absolute Length Units

px	Pixels.
pt	Points.
cm	Centimeters.
mm	Millimeters.
in	Inches.

Relative Length Units

<code>%</code>	Size dependent on parents.
<code>em</code>	Relative to the font-size of the element.
<code>rem</code>	Relative to the font-size of the root element.
<code>vw</code>	Relative to the width of the viewport.
<code>vh</code>	Relative to the height of the viewport.

Colors can be specified in several ways:

<code>red</code>	By a color word.
<code>#f00</code>	Using hexadecimal notation for (#RGB).
<code>#ff0000</code>	Using hexadecimal notation (#RRGGBB).
<code>rgb(255, 0, 0)</code>	Using RGB values (0–255).
<code>rgba(255, 0, 0, 1.0)</code>	Using RGB values and alpha (0.0—1.0).
<code>hsl(240, 100%, 50%)</code>	Hue (0–255), Saturation (%), Lightness (%).

Display property

The `display` property indicates how an element will be displayed

`none` Not shown at all (invisible).

`block` As a box

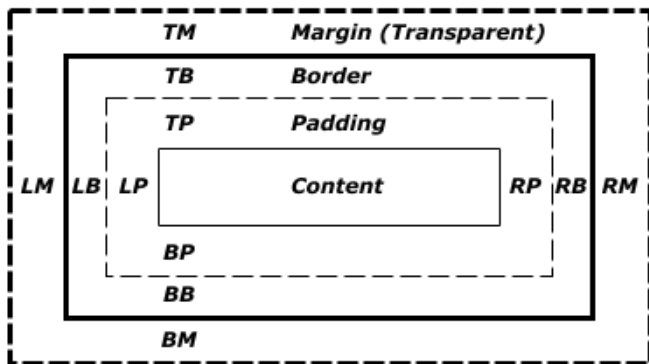
`inline` In sequence with line elements





`inline-block` As an inline box

`flex` Using the Flexbox algorithm

`inline-flex` An inline box using the Flex algorithm.

CSS Box Model



-  Margin edge
-  Border edge
-  Padding edge
-  Content edge

Margin and Padding

`margin` and `padding` set the lengths in this order:

top — right — bottom — left.

```
div.my-box {  
  margin: 1rem .5rem .8rem .5rem;  
  padding: 10px 5px 8px 6px;  
}
```

Omitted values

When we omit values, they are copied in a common-sense way:

```
.all-equal {  
  margin: 1rem; /* top = right = bottom = left */  
}  
.symmetric {  
  padding: .3rem .6rem; /* top = bottom, left = right */  
}  
.left-right {  
  margin: .3rem .6rem .1rem; /* left = right */  
}
```

Specific values

Specific properties let us change specific values: `margin-top` ,
`margin-right` , ...

```
p {  
  margin: 0;  
  margin-left: 1rem;  
}  
h1 {  
  padding-top: 0;  
}  
ul {  
  padding-left: .5rem;  
}
```

Combined properties

margin :

-top, -right, -bottom, -left

padding :

-top, -right, -bottom, -left

font :

-style, -weight, -size, -family.

background :

-color, -image, -position, -repeat, -attachment, -origin, -clip,
-size.

Text properties

color

The color of the text.

line-height

The height of one line (the interline space).

letter-spacing

Space between letters in a text.

word-spacing

Space between words in a text.

text-align

center | left | right | justify.

text-decoration

underline | line-through.

text-transform

uppercase | lowercase | capitalize.

text-indent

Indentation on the first line of a paragraph.

Font Properties

The `font` property admits many fonts as fallbacks (separated by commas).

```
h1 {  
  font: italic 30px Verdana, 28px Helvetica, sans-serif;  
}
```

Specific properties:

`font-family` The name of the font to use.

`font-size` Size of the font (in px, em, rem, or %).

`font-weight` Weight of the font (100 to 900, bold | normal)

`font-style` italics | oblique.

Backgrounds

The property `background-color` changes the background of an element:

```
body { background-color: #ccc; } /* grey background */  
li a {  
    background-color: blue; /* blue background */  
    color: white;          /* with white text */  
}
```

With `background-image` an image can occupy the background:

```
div.chess {  
    background-image: url(images/checkerboard.jpg);  
}
```

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

The Cascade

Ambiguous rules

The same element can be the target of rules with different values

```
h1 { color: red; }  
body h1 { color: green; }
```

```
h2.grape { color: purple; }  
h2 { color: silver; }
```

```
html > body table tr[id="totals"] td ul > li { color: maroon; }  
li#answer { color: navy; }
```

How do we know which one will win?

Specificity

A selector's specificity is determined by its components.

Specificity is a vector with 4 components: **(0, 1, 5, 2)**.

Components to the left of the vector weight more.

$(0, 0, 2, 0) > (0, 0, 1, 0)$

$(0, 1, 0, 0) > (0, 0, 1, 0)$

The actual specificity is determined **adding up** the different contributions:

For every element add **(0, 0, 0, 1)**

For every class add **(0, 0, 1, 0)**

For every ID add **(0, 1, 0, 0)**

For every inline style add **(1, 0, 0, 0)**

Specificity examples

<code>h1 { color: purple; }</code>	<code>/* 0, 0, 0, 1 */</code>
<code>p em { color: blue; }</code>	<code>/* 0, 0, 0, 2 */</code>
<code>.grape { color: green; }</code>	<code>/* 0, 0, 1, 0 */</code>
<code>*.bright { color: yellow; }</code>	<code>/* 0, 0, 1, 0 */</code>
<code>p.bright e.dark { color: black; }</code>	<code>/* 0, 0, 2, 2 */</code>
<code>#id216 { color: orange; }</code>	<code>/* 0, 1, 0, 0 */</code>
<code>div#sidebar *[href] { color: pink; }</code>	<code>/* 0, 1, 1, 1 */</code>
<code>nav#menu li#top { color: red; }</code>	<code>/* 0, 2, 0, 2 */</code>

Importance

The `!important` keyword marks a rule as being above all others.

```
p.dark {  
  color: #333 !important;  
  background: white;  
}
```

`!important` *must go just before the semicolon.*

Important declarations have the same specificity but are considered separately to others. They always win against non-important declarations.

The ! sign does not mean negation as in many programming languages!

Inheritance

Some rules apply to an element *and its descendants*:

- `color` ,
- `font-size` ,
- `font-family` ...

Some rules apply just to the root element:

- `border` ,
- `margin` ,
- `margin` ...

Which rules apply or not to descendants is down to common sense.

1. Importance

If the rule was marked as `!important` (also transitions and animations).

2. Origin

Where the rule was defined (website, user, browser defaults).

3. Specificity

The specificity vector explained before.

4. Order

The order of declaration (last rule overwrites a previous one).

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

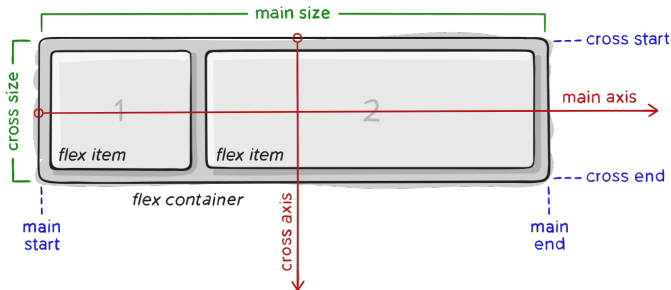
Flexbox

Flexbox

Flexbox is a new mechanism to align items within its parent.

To use the Flexbox algorithm you have set `display` to `flex`:

```
div.carousel { display: flex; }
```



Flexbox Main Parameters

A component can specify the *layout of its children* with 3 main parameters.

flex-direction

Specify the **primary axis**

`column` (default) `row`

justify-content

Distribution of children along the **primary axis**

`flex-start` `center` `flex-end`

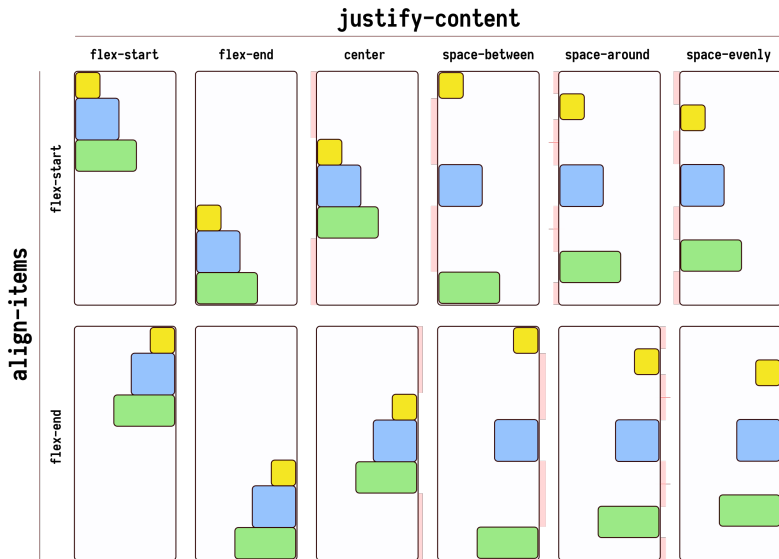
`space-around` `space-between` `space-evenly`

align-items

Alignment of children along the **secondary axis**

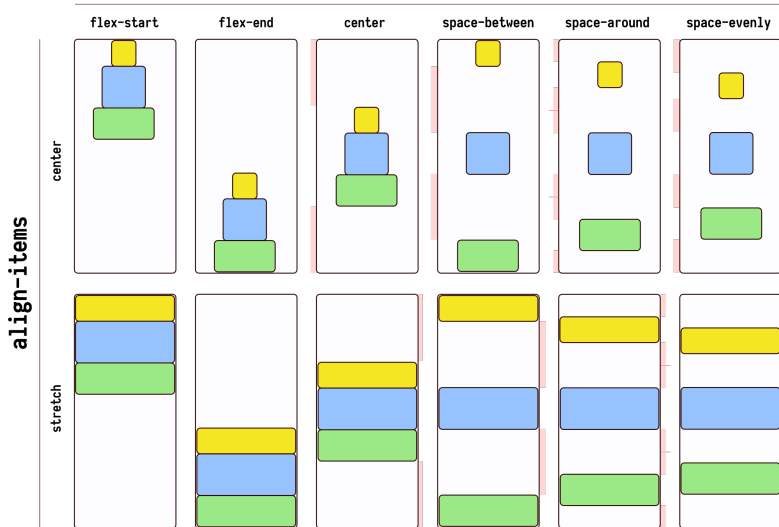
`flex-start` `center` `flex-end` `stretch`

Flexbox Table 1



Flexbox Table 2

justify-content



Component Dimensions

A component can either have fixed dimensions, or flex dimensions.

Fixed Dimensions

Specified using `width` and `height`, both in DIP (Device Independent Pixels).

Flex Dimensions

Specified using `flex`, which is a "stretch factor". The element will expand and shrink dynamically based on available space, according to the value of `flex`.

A Flexbox Game

A game for learning flexbox!

FLEXBOX FROGGY



Welcome to Flexbox Froggy, a game where you help Froggy and friends by writing CSS code! Guide this frog to the lilypad on the right by using the `justify-content` property, which aligns items horizontally and accepts the following values:

- **flex-start**: Items align to the left side of the container.
- **flex-end**: Items align to the right side of the container.
- **center**: Items align at the center of the container.
- **space-between**: Items display with equal spacing between them.
- **space-around**: Items display with equal spacing around them.

For example, `justify-content: flex-end;` will move the frog to the right.

```
1 #pond {
2   display: flex;
3
4 }
5
6
7
8
9
10
```

Next



Flexbox Froggy is created by [Coderojo](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

Want to learn CSS grid? Play [Grid Garden](#)

<https://flexboxfroggy.com/>

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

Pseudo classes/elements

Pseudo-classes refer to classes that we don't have to declare, and which represent stuff that the browser already knows about:

- If an element is empty.
- Which position an element is (first, last, ...).
- The state of a link (enabled, disabled, clicked, visited, focused, ...).
- The validity of form controls (valid, invalid).
- Negation of other selector (not).

Pseudo-elements refer to parts of elements or special elements:

- First letter of line.
- Before/After content.

Empty elements

The `:empty` class refers to elements which have no content

```
<p>A paragraph with text</p>  
<p></p>
```

```
p:empty { margin: 0; }
```


First/last child

The `:first-child` class applies to elements that occupy the first position in the list of children.

The `:last-child` is analogous for the last child.

```
li:first-child { border-top: 1px solid #ccc; }  
li:last-child  { border-bottom: 1px solid #ccc; }
```

Child in position N

The `nth-child(n)` lets you use an index or a formula:

```
li:nth-child(2) {  
  background-color: yellow;  
}  
tr:nth-child(2) td:nth-child(3) {  
  border-color: gray;  
}  
ul li:nth-child(2n) {  
  text-transform: uppercase;  
}
```

Link state classes

- `:link` : An unvisited link.
- `:visited` : An visited link.
- `:focus` : A link in focus.
- `:hover` : A link with cursor on top.
- `:active` : A link being clicked.

Negation pseudo-class

The `:not(sel)` class is true for elements that aren't `sel`.

`sel` *es un selector simple (tipos, clases sin jerarquía).*

```
li:not(.more-info) {  
  color: red;  
}  
*:not(section) > table {  
  display: none;  
}  
.link:not(li):not(p) {  
  font-style: italic;  
}
```

Before/After pseudo-elements

`::before` and `::after`, in combination with the `content` property, let you add prefixes and suffixes:

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

Media Queries

Media Queries

@media queries are about applying **different CSS** as a function of the medium you are in.

There are numerous reasons to change the style depending on the device we are on:

- Phones are **narrow** and desktops **wide**.
- Some devices have high **pixel density**.
- Some devices cannot do "**hover**" (since there is no mouse).
- Devices can change the **orientation**.
- A **printed** document should have different styles.

The @media rule

@media specifies a condition and a list of rules to apply if that condition is valid.

```
@media condition {  
    /* ...  
       rules to apply if condition is true  
    ... */  
}
```

To combine different conditions, operators **and**, **or** and **not** are available.

Media types

These basic expressions are true in different environments:

- `all`, true for all environments.
- `screen`, only true for screens.
- `print`, only true when printing.

```
body {  
  background-color: darkblue;  
}  
@media print {  
  body {  
    background-color: white;  
  }  
}
```

Media width

To consider media width:

- `(min-width: W)`, true if width is at least W .
- `(max-width: W)`, true if width is at most W .

```
@media screen and (max-width: 400px) {  
  .avatar { width: 2em; height: 2em; }  
}  
@media screen and (min-width: 401px) {  
  .avatar { width: 3em; height: 3em; }  
}
```

Usual mobile, tablet and desktop schemes

To treat mobiles, tablets, and desktops differently a scheme similar to this can be used:

```
@media screen and (max-width: 400px) {  
    /* mobile */  
}  
@media screen and (min-width: 401px) and (max-width: 768px) {  
    /* tablet */  
}  
@media screen and (min-width: 769px) {  
    /* desktop */  
}
```

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

Vendor Prefixing

Vendor prefixing

Some CSS properties are *browser dependent*.

Some browsers added new features before other browsers.

By using vendor prefixes, some websites could use those features.
(*This became popular with CSS3*)

Prefixes:

- **-moz-** : Mozilla (Firefox).
- **-webkit-** : Webkit based browsers (Safari, Chrome).
- **-o-** : Opera.
- **-ms-** : Microsoft IE or Edge.

Vendor prefixes example

```
.main-sidebar {  
  width: 20;           /* For old syntax */  
  flex: 1;             /* New spec */  
  -webkit-box-flex: 1; /* OLD: iOS 6-, Safari 3.1-6 */  
  -webkit-flex: 1;     /* Safari 6.1+. iOS 7.1+, BB10 */  
  -ms-flex: 1;         /* IE 10 */  
}
```




Autoprefixer is a plugin for PostCSS (a CSS Postprocessor) that automatically adds prefixes to your CSS.

<https://github.com/postcss/autoprefixer>

Table of Contents

CSS

Syntax

Selectors

Values & Properties

The Cascade

Flexbox

Pseudo classes/elements

Media Queries

Vendor Prefixing

CSS Libraries & Tools

CSS Libraries & Tools

<https://sass-lang.com/>


Sass just launched a brand new module system. [Learn all about the module system on the Sass blog!](#)

[Install](#)[Learn Sass](#)[Blog](#)[Documentation](#)[Get Involved](#)

CSS with superpowers

Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.



Current Releases: [Dart Sass 1.25.0](#) [LibSass 3.6.3](#) [Ruby Sass](#)  [Implementation Guide](#)

CSS Compatible

Sass is completely compatible with all versions of CSS. We take this compatibility seriously, so that you can seamlessly use any available CSS libraries.

Feature Rich

Sass boasts more features and abilities than any other CSS extension language out there. The Sass Core Team has worked endlessly to not only keep up, but stay ahead.

Mature

Sass has been actively supported for about 13 years by its loving Core Team.

<https://getbootstrap.com/>



[Home](#) [Documentation](#) [Examples](#) [Icons](#) [Themes](#) [Expo](#) [Blog](#)

v4.4



[Download](#)

Bootstrap

Build responsive, mobile-first projects on the web with the world's most popular front-end component library.

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

[Get started](#)

[Download](#)

Currently v4.4.1



Installation

Include Bootstrap's source Sass and JavaScript files via npm, Composer or Meteor. Package managed installs don't include documentation, but do include our



BootstrapCDN

When you only need to include Bootstrap's compiled CSS or JS, you can use [BootstrapCDN](#).



Official Themes

Take Bootstrap 4 to the next level with official premium themes—toolkits built on Bootstrap with new components and plugins, docs, and build tools.

<https://bulma.io/>

[Documentation](#)[Videos](#)[Expo](#)[Love](#)[Backers](#)[More](#)[Sponsor](#)[New!](#)[BECOME A PATRON](#)[Download](#)

Bulma is a free, open source CSS framework based on **Flexbox** and used by more than **200,000** developers.



38,510

downloads

666k/month

```
npm install bulma
```

[copy](#)[Download v0.8.0](#)[View docs](#)

Video by Vue Mastery

100% Responsive

Designed for mobile first

Modular

Just import what you need

Modern

Built with Flexbox

Free

Open source on GitHub