CSS

Marco Torchiano

Version 2.3.1 - April 2020

License

Basics

CSS

Cascading Style Sheets (CSS) is a language for specifying how documents are presented

CSS enables the separation of:

- Contents (HTML)
- Formatting / presentation (CSS)

Advantages

- Avoid duplication:
  - formatting is written once and applied uniformly to the all pages in the site
- Easier maintenance
  - there is one source of truth for the presentation: any change to the CSS style automatically propagates to all the linking documents
- Reuse
  - same content can be shown with different styles

CSS working

CSS defines a set of rules.

Each rule is made up of:

- **selector**: identifies which HTML elements are targeted
- **declaration block**, contains several *declarations* each consisting of
  - **property**: specifies the feature (e.g. color, size, font, etc.) of the styled HTML elements
  - **value**: defines the value to control the appearance of the feature for the selected element(s)

CSS Rules

```
selector {
  property1: value1;
  property2: value2;
  /* ... */ }
```

- the *selector* selects the elements the rule applies to,
- each declaration define the style:
  - the *properties* are presentation features,
  - the *values* are the options/values of the features.

Basic Selectors

- **Element** type selectors,
  - e.g. `p`

    selects all elements of the given type
- **Class** selectors,
  - e.g. `.example`

    selects all elements with the given class
- **ID** selectors,
  - e.g. `#warning`

    selects the individual elements with the given id

IDs and Classes

- Classes are defined by the `class` attribute

  A class can be applied to multiple elements and any element can belong to several classes.

  `<p class="quote high">`

- IDs are defined by the `id` attribute of an element

  They are meant to be unique and single.

  `<p id="warning">`

Selector Compositon

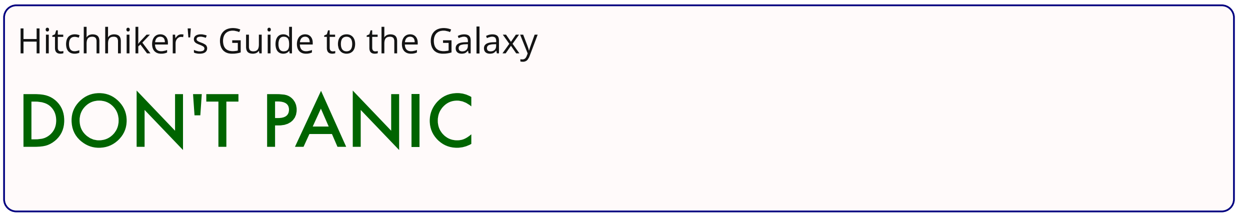
- alternatives: `,` separate alternative selectors for which the rule applies, e.g.
  - `p, li` applies to both `<p>` and `<li>` elements
- combination: sequence of collated selectors define the joint conditions for applying the rule, e.g.
  - `p.warning` applies to all `<p>` elements with class `warning`
  - `p#quote` applies to the `<p>` element with id `example`

Application of CSS to HTML

- Embedded
  - using the html `style` element
- External
  - using a html `link` element referring to an external style sheet
- Inline
  - using the `style` attribute of html elements
  - contains only the declarations
  - the selector is implied (the current element)

Embedded style with id

```
<style type="text/css">
  p#warning {
    color:darkgreen;
    font-size:200%;
    font-family: futura;
  }
</style>
<p>Hitchhiker's Guide to the Galaxy</p>
<p id="warning">DON'T PANIC</p>
```

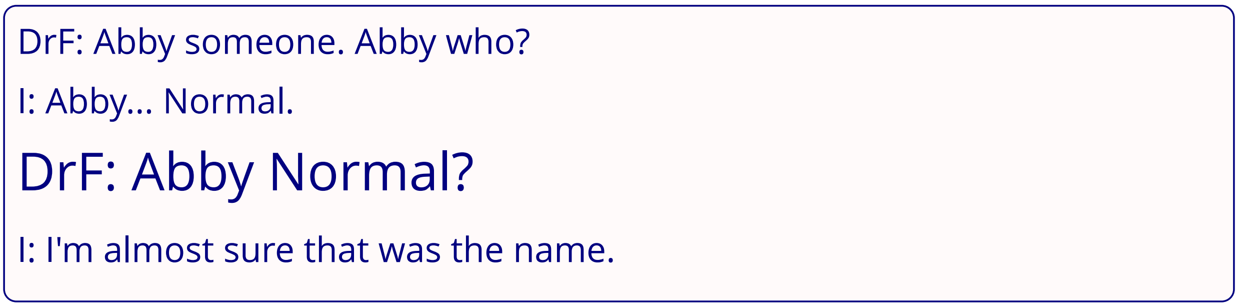


Hitchhiker's Guide to the Galaxy

DON'T PANIC

Embedded style with class

```
<style type="text/css">
  p.l { color:navy; }
  p.high { font-size:150%; }</style>
<p class="l">DrF: Abby someone. Abby who?</p>
<p class="l">I: Abby... Normal.</p>
<p class="l high">DrF: Abby Normal?</p>
<p class="l">I: I'm almost sure that was the name.</p
```



DrF: Abby someone. Abby who?

I: Abby... Normal.

DrF: Abby Normal?

I: I'm almost sure that was the name.

External style

The `head` element can include a `link` element that refers to a separate resource:

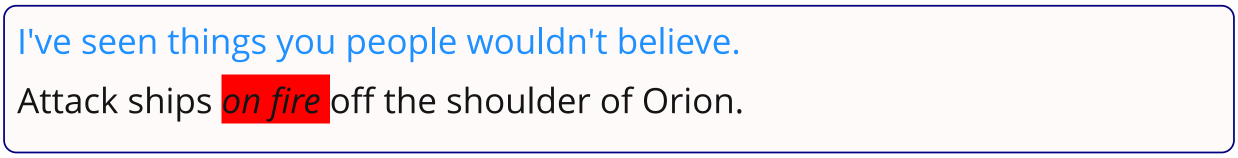
```
<link rel="stylesheet" href="styles.css" >
```

External styles can be imported from an embedded style

```
<style type="text/css">
  @import url("styles.css");
  /* ... */
</style>
```

Inline style

```
<p style="color:dodgerblue;">
I've seen things you people wouldn't believe.
</p>
<p>
Attack ships <em style="background-color:red;">
on fire </em> off the shoulder of Orion.
</p>
```



I've seen things you people wouldn't believe.

Attack ships *on fire* off the shoulder of Orion.

Style Attributes

Color

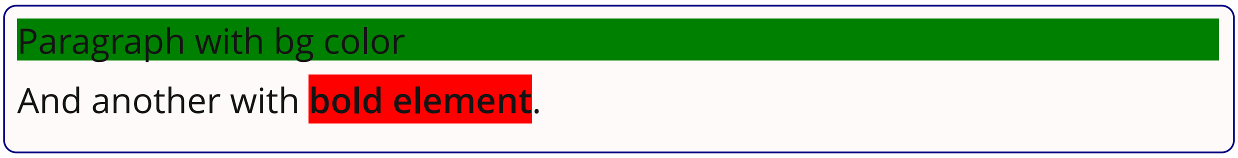
- `color` defines the text color
- `background-color` or just `background` defines the background color of the element
    - Note: block elements (e.g. `<p>`) usually span the full width of the page.

Color can be define as:

- name
- rgb
- hsl

Background Color

```
<p style="background:green;">
Paragraph with bg color</p>
<p>And another with
<b style="background:red">bold
element</b>.</p>
```



Paragraph with bg color

And another with **bold element**.

Color Names

- Name
  - CSS has a set of predefined names,

    see: https://www.w3.org/TR/css-color-3/#svg-color
  - The basic standard names supported are:

aqua	black	blue	fuchsia
gray	green	lime	maroon
navy	olive	purple	red
silver	teal	white	yellow

Color RGB

- Levels of **R**ed **G**reen and **B**lue

  can be specified as
  - `rgb(0,0,0)` with each number (between 0 and 255) indicating the R, G, and B intensity

    rgb(255,0,0) rgb(0,255,0) rgb(0,0,255)
  - hex notation `#000000`, where each pair of digits represents the value of one component

    #ff0000 #00ff00 #0000ff

Color HSL

- Levels of **H**ue, **S**aturation, and **L**ightness
  - hue is a degree on the color wheel from 0 to 360: 0 is red, 120 is green, 240 is blue.
  - saturation is a percentage where 0% is a gray and 100% is fully saturated color
  - lightness is a percentage where 0% is black and 100% is white

  can be specified as `hsl(0,0%,0%)`

hsl(0,100%,50%) hsl(120,100%,50%) hsl(240,100%,50%)

Color wheel

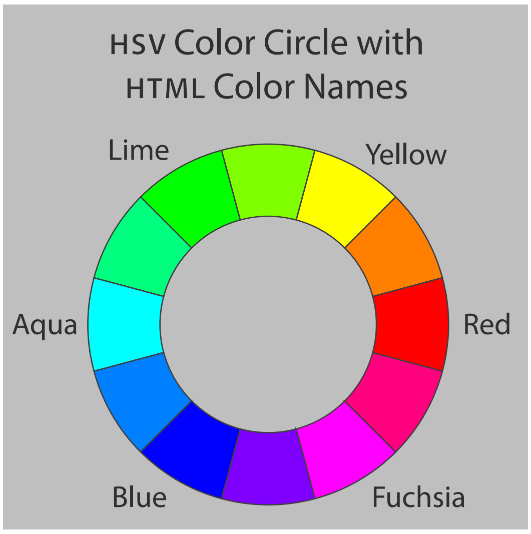


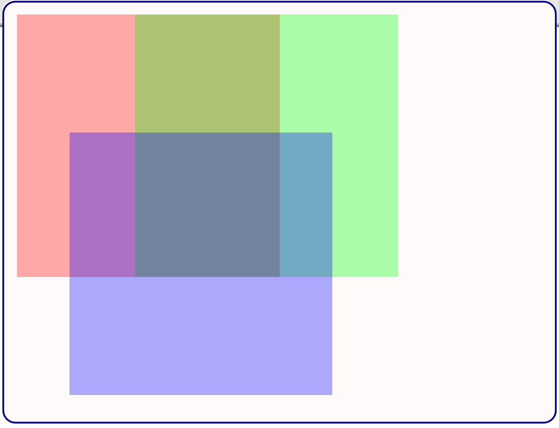
Image cc-by-sa by Jacobolus

Transparency

- Both RGB, and HSL can have an additional parameter, *alpha*; a number between 0.0 (fully transparent) and 1.0 (fully opaque).
  - the functions are `rgba()` and `hsla()`
  - full transparent color is named `transparent`

Transparency: example

```
<div style="background-color:rgba(255,0,0,.33);">
    </div>
<div style="background-color:rgba(0,255,0,.33);
    left:100px;"></div>
<div style="background-color:rgba(0,0,255,.33);
    left:50px;top:100px;"></div>
```



Border

Defines the border of the element

- `border-width`: width, typically in `px`
- `border-style`: type of the line
  - can be: `none`, `solid`, `dotted`, `dashed`, `double`
- `border-color`: any color
- `border-radius`: radius of the vertex connections

Border

A compact form is possible that lists all the elements separated by spaces:

```
border: 2px solid black;
```

is equivalent to:

```
border-width: 2px;
border-style: solid;
border-color: black;
```

Size Units

Absolute sizes:

- `in`, `cm`, `mm`: represent the corresponding lenght
- `px`: represents a screen pixel, or *display unit* (1/96 of inch or 0.26 mm @ 71 cm distance)
  - Note: a CSS pixel (px) **is not** a device pixel!
- `pt`: represents a screen point (1/72 of inch)

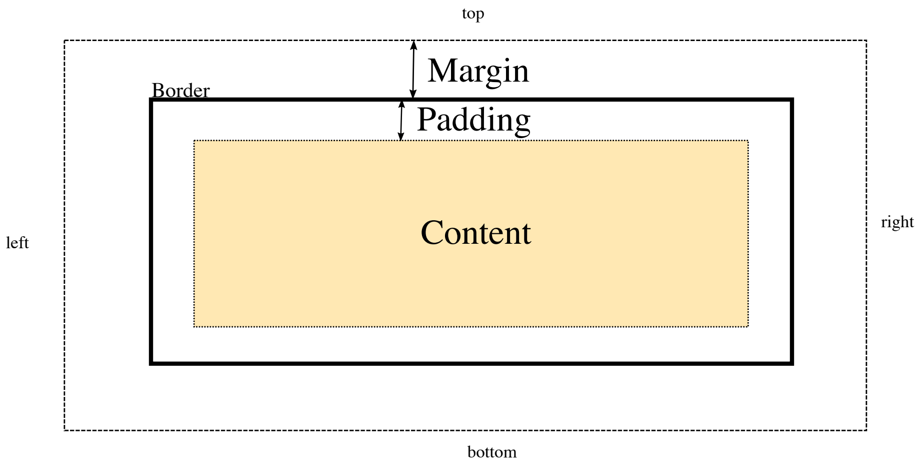
Relative sizes:

- `%` relative size w.r.t. the parent/container element

Units

Unit	Description
em	the computed value of the font-size on the element on which it is used.
ex	the current font’s x-height, equal to the height of a lowercase ‘x’
ch	the advance of the ‘0’ (zero) glyph in the current font, `ch` stands for character.
rem	the computed value of the font-size property for the document’s root element.

Box Model



Box Model

Box Model

Border, Margin, and Padding features can be expressed:

- overall: same value applies to all four sides of box
- separately for each side: `top`, `right`, `bottom`, `left`
  e.g. `margin-top: 26pt;`

  The values in clockwise order from the top:
  e.g. `border-width: 1pt 2pt 4pt 8pt;`

  Two values meaning top&bottom, left&right

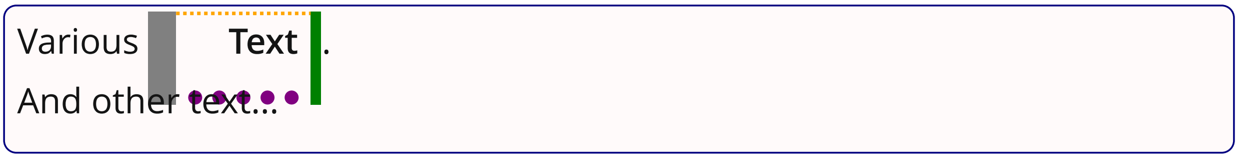
  For the border the side is indicated before the feature:
  e.g. `border-left-width:2pt;`

Display and Visibility

- `display`:
  - `none` prevents rendering the element,
  - `block` forces line break,
  - `inline` placed in line, size is given by content,
  - `inline-block` allow for width and height and vertical spacing.
- `visibility`:
  - `hidden` not shown though occupies the relative space in the page
  - `visible` can be seen

Box Model Example

```
<p>Various <b style=
 "border-width: 2pt     6pt    8pt   16pt;
              /*top    right bottom left */
  border-color: orange green purple gray;
  border-style: dotted solid;
  padding: 0px 10px 20px 40px;">Text</b>.</p>
<p>And other text...</p>
```

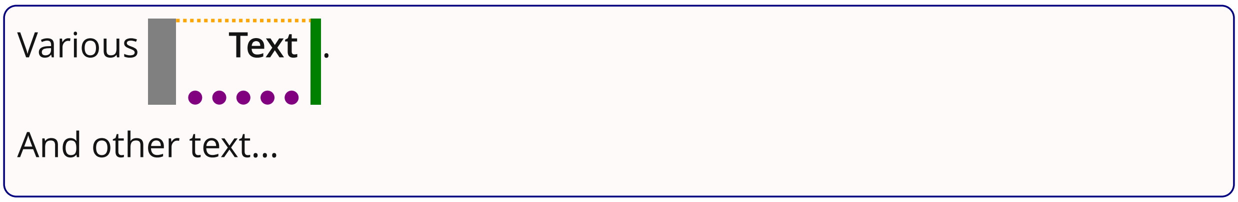


Various **Text**.

And other text...

Box Model Example with Inline-block

```
<p>Various <b style=
 "display: inline-block;
  border-width: 2pt     6pt    8pt  16pt;
              /*top    right bottom left */
  border-color: orange green purple gray;
  border-style: dotted solid;
  padding: 0px 10px 20px 40px;">Text</b>.</p>
<p>And other text...</p>
```



Various Text.

And other text...

Size

`height` and `width` define the size of the element. Valid values are:

- `auto` : size is computed by the browser,
- absolute size as number with unit,
- a `%` relative to the container element.

`max-width` defines the maximum width, the element is resized when the window is smaller than the specified value.

- `none` means there is no max width

Text

- `text-align`: `left`, `right`, `center`, `justify`
- `text-decoration`: `none`, `underline`, `overline`, `line-through`
- `text-transform`: `uppercase`, `lowercase`, `capitalize`
- `text-indent`: *`length`*
- `line-height`: `normal`, *`number`*, *`length`* , `%`

Font

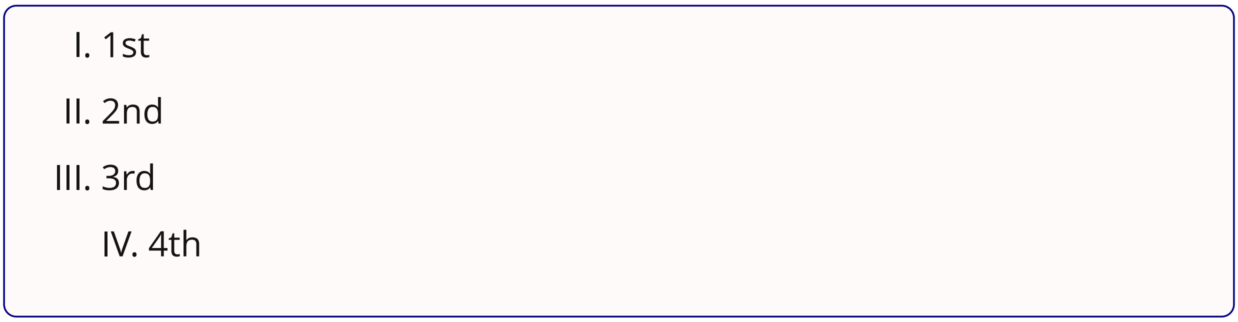
- `font-family`: `serif`, `sans-serif`, or font name, e.g. “Times New Roman”
- `font-style`: `normal`, `italic`, `oblique`
- `font-variant`: `normal`, `small-caps`
- `font-weigth`: `normal`, `bold`, `bolder`, `lighter`, *`number`* (400 is nomal, 700 is bold)
- `font-size`: `medium`, `xx-small`, `x-small`, `small`, `large`, `x-large`, `xx-large`, `smaller`, `larger`, *`length`*, `%`

Lists

- `list-style-type`
    - unordered `<ul>`: `disc`, `circle`, `square`, `none`
    - ordered `<ol>`: `decimal`, `lower-alpha`, `lower-roman`, …
- `list-style-position`
    - `outside` (default) outside the `li` element
    - `inside` inside the `li` element, takes space
- `list-style-image`: specifies an image with `url('image.gif')`

Example list

```
<ol style="list-style-type: upper-roman;
           padding-left: 1em;">
<li>1st</li> <li>2nd</li> <li>3rd</li>
<li style="list-style-position: inside;">4th</li>
</ol>
```



I. 1st
II. 2nd
III. 3rd
IV. 4th

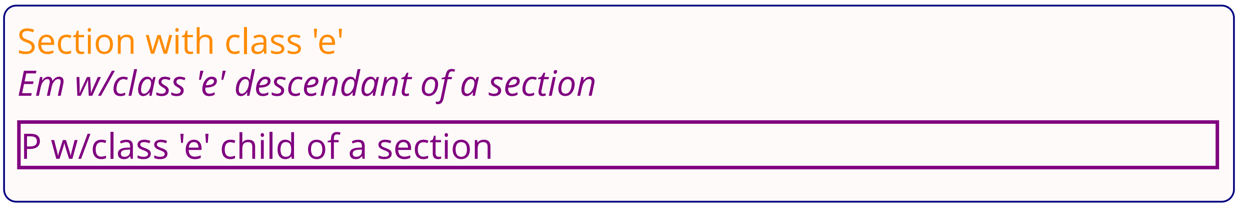
Selectors

Selector combination and relationship

Selector	Selects
`A E`	Any `E` element that is a *descendant* of an `A` element
`A > E`	Any `E` element that is a *child* of an `A` element
`B + E`	Any `E` element that is the *next sibling* of a `B` element
`B ~ E`	Any `E` element that is *any sibling* of a `B` element

Example selectors

```
<style> section.e { color:darkorange; }
        section .e { color: purple; }
        section > .e { border: 2pt solid; }
</style>
<section class='e'>Section with class 'e'</section>
<section><p><em class="e">Em w/class 'e' descendant
                          of a section</em></p>
<p class='e'>P w/class 'e' child of a section</p>
</section>
```



Section with class 'e'

Em w/class 'e' descendant of a section

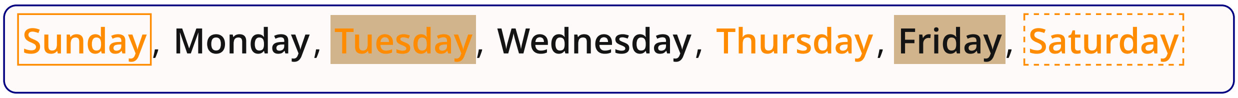
P w/class 'e' child of a section

Childrens

Selector	Selects
`E:first-child`	Any `E` element that is the first child of its parent
`E:last-child`	Any `E` element that is the last child of its parent
`E:nth-child(n)`	Any `E` element that is the nth child of its parent (*n* can be a number, `odd`, `even`, or any expression, e.g. `2n+1`)
`E:only-child`	Any `E` element that is the only child of its parent

Example selectors

```
<style>
.c :first-child   { border: 1pt solid;}
.c :nth-child(odd){ color:darkorange; }
.c :nth-child(3n) { background-color: tan; }
.c :last-child    { border: 1pt dashed;}
</style>
<p class="c">
<b>Sunday</b>, <b>Monday</b>, <b>Tuesday</b>,
<b>Wednesday</b>, <b>Thursday</b>, <b>Friday</b>,
<b>Saturday</b>
</p>
```



Sunday, **Monday**, **Tuesday**, **Wednesday**, **Thursday**, **Friday**, **Saturday**

Pseudo classes

Selector	Selects
`:link`	Normal state of a link
`:visited`	Links that have already been visited
`:active`	Links that are being clicked
`:hover`	Elements (e.g. links but not only) that are hovered on

Example selectors

```
<style> .p :link   { color: orange;}
        .p :visited{ color: gray; }
        .p :hover  { background-color: yellow;}
</style>
<p class="p">
There are <a href="http://www.polito.it">visited
links</a>, <a href="exist.not">non visited links
</a>, and <em>non links</em> children</p>
<p class="p">Text is not a child but
<fake>fake is</fake>.</p>
```

There are visited links, non visited links , and *non links* children

Text is not a child but fake is.

Attribute selectors

Selector	Selects
`E[attr]`	Any `E` with an attribute `attr`
`E[attr="val"]`	Any `E` with an attribute `attr` having a value `"val"`
`E[attr^="beg"]`	Any `E` with an attribute `attr` having a value starting with `"beg"`
`E[attr$="end"]`	Any `E` with an attribute `attr` having a value ending with `"end"`
`E[attr*="sub"]`	Any `E` with an attribute `attr` having a value containing `"sub"`

Cascade

Sources

CSS declaration for style attributes may origin from different sources:

- User-agent stylesheets
  - defined in the browser, provide default styles
- Author stylesheets
  - defined by web developers
- User styleshees
  - might be defined by the user to customize user experience

Specificity

Specificity is a priority assigned to rules, according to the following order:

1. `style` attribute of element
2. `#id` selector
3. attribute selector (including `class`)
4. element name selector

Cascade

The cascading algorithm determines how to find the value to apply for each property for each document element.

Rules are ordered by:

- `!important` specifier
- source (user,author,user-agent)
- specifity
- order of appearance (last declaration first)

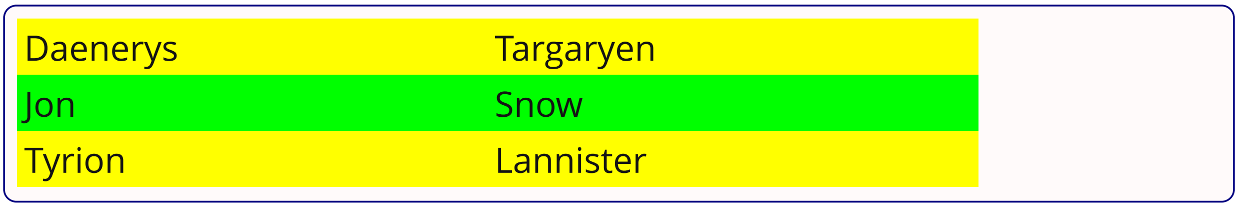
Advanced Attributes

Tables

- borders can be set for `table`, `th`, `td` or a subset of them
  - `border-collapse`: if `collapse` merges adjacent borders
- `height` and `width` define the size of the table

Example Tables

```
<style>
.extab tr{ background-color: lime;}
.extab tr:nth-child(odd){
      background-color: yellow;}</style>
<table class="extab">
    <tr><td>Daenerys</td><td>Targaryen</td></tr>
    <tr><td>Jon</td><td>Snow</td></tr>
    <tr><td>Tyrion</td><td>Lannister</td></tr>
</table>
```



Daenerys	Targaryen
Jon	Snow
Tyrion	Lannister

Floating

Elements can float

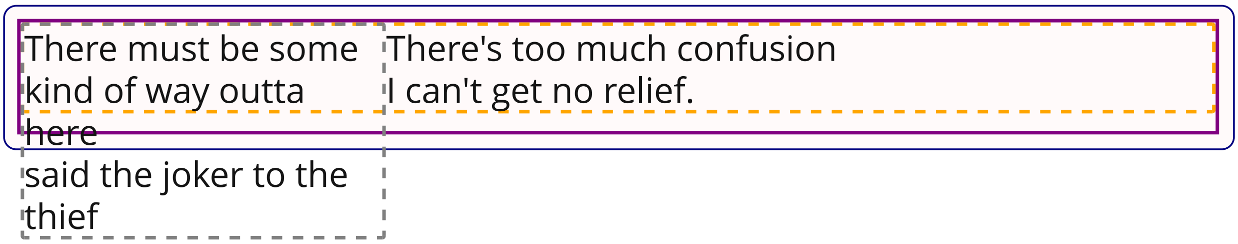
- `float`: `none`,`left`,`right`
- `clear`: `none`, `left`, `right`, `both`
  On which side of the element other text is *not* allowed to wrap

Note: a floating element is not used to compute the container size, so it can *overflow* the container.

- `overflow` : `auto` adapts the container size including floating elements

Example Float

```
<section style="border:solid purple;">
<p style="width:30%;float:left;border:dashed gray">
There must be some kind of way outta here<br>
said the joker to the thief</p>
<p style="border:dashed orange;">
There's too much confusion<br>
I can't get no relief.</p></section>
```

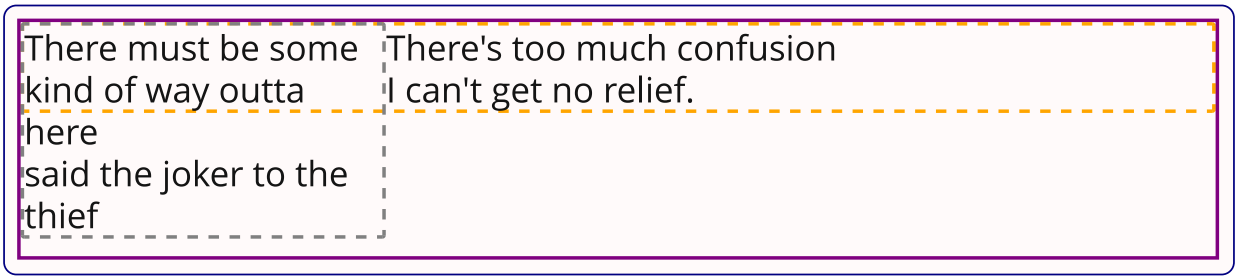


There must be some kind of way outta here
said the joker to the thief

There's too much confusion
I can't get no relief.

Example Float Overflow

```
<section style="border:solid purple;overflow:auto">
<p style="width:30%;float:left;border:dashed gray">
There must be some kind of way outta here<br>
said the joker to the thief</p>
<p style="border:dashed orange;">
There's too much confusion<br>
I can't get no relief.</p></section>
```



There must be some kind of way outta here
said the joker to the thief

There's too much confusion
I can't get no relief.

Position

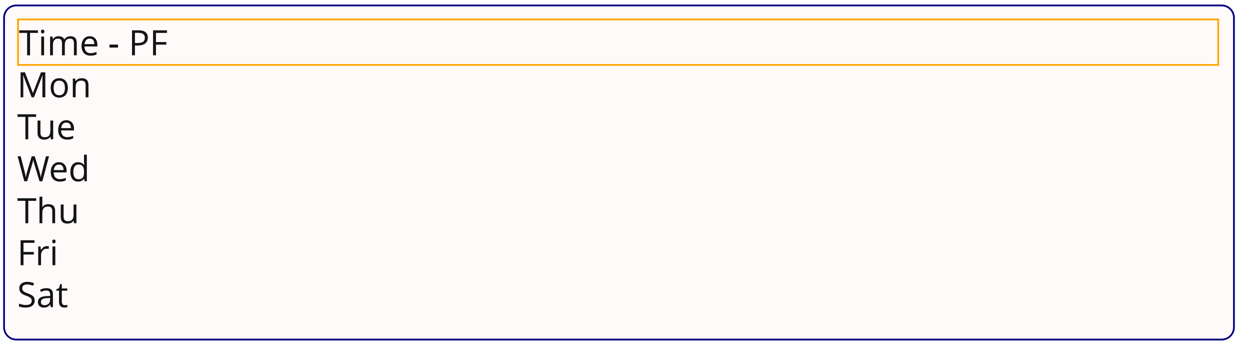
- `position` defines the positioning strategy
  - `static`: follow the page flow
  - `relative`: placed at offset from normal position
  - `fixed`: placed at offset w.r.t. viewport
  - `absolute`: placed at offset w.r.t. position ancestor
  - `sticky`: like *relative*, switch to *fixed* when reaching margin
- the actual offset is specified using `top` `bottom` `left` `right`

Example: header (css)

```
<style>
.myheader {
    position: absolute;
    top:0px;
    left:0; right:0; /* full width */
    margin-top:0pt;
    height: 25pt;
    border: 1pt orange solid;
}
.mybody {
    position: relative;
    padding-top: 25pt;
}
</style>
```

Example: header (html)

```
<body. class="mybody">
<p>Mon<br>Tue<br>Wed<br>Thu<br>Fri<br>Sat</p>
<p>Jan<br>Feb<br>Mar<br>Apr<br>May<br>Jun</p>
<header class="myheader">Time - PF</header>
</body.>
```



```
Time - PF
Mon
Tue
Wed
Thu
Fri
Sat
```

References

Online resources

- MDN - CSS: Cascading Style Sheets
  https://developer.mozilla.org/en-US/docs/Web/CSS
- W3C CSS home page
  https://www.w3.org/Style/CSS/Overview.en.html
- W3C CSS Tutorials
  https://docs.webplatform.org/wiki/css/tutorials
- W3Schools Tutorial on CSS
  http://www.w3schools.com/css/default.asp