CSS selectors

Patterns used to select elements to style. Selectors can override each other going from least to most “specific,” element < class < id < inline.

`element` selectors

Styles elements of a certain type, such as paragraphs or headings.

Code example:

```
p {
  background-color: yellow;
}
```

`.class` selectors

Styles elements with a class attribute. Elements can be given multiple classes, and classes can be used throughout the page to style similar elements.

Code example:

```
.intro {
  color: #00ff00;
  background-color: #e6e6e6;
}
```

`#id` selectors

Styles the element with the specified id. Ids should be unique and each only assigned to one element on the page.

Code example:

```
#firstname {
  color: rgb(255, 255, 0);
}
```

Inline style

Elements can also be styled with an inline style attribute

Code example:

```
My mother has <span style="color:blue">blue</span> eyes.
```

`font-family`

Specifies the font, can hold several font names as a “fallback.” The generic font names are sans-serif, serif, fantasy, monospace, and cursive.

Code example:
```
body {
  font-family: "Times New Roman", serif;
}
```

`font-size`

Sets the font size, in pixels, percentage, or relative sizing.

Code examples:
```
body {
  font-size: 12px;
}
h1 {
  font-size: 2em;
}
```

`font-weight`

Sets characters to degrees of thickness. 400 is normal, and 700 is the same as bold.

Code examples:
```
h1 {
  font-weight: bold;
}
h2 {
  font-weight: 700;
}
```

`font-style`

Can be used to sets the font style to italic or normal.

Code example:
```
p {
  font-style: italic;
}
```

`text-decoration`

Decoration added to text

Code example:
```css
h3 {
  text-decoration: underline;
}
a {
  text-decoration: none;
}
```

`text-align`

Sets the horizontal alignment of text in an element.

Code example:
```css
.username {
  text-align: right;
}
```

`line-height`

Sets the height of a line in pixels, percentage, or relative sizing

Code example:
```css
h1 {
  line-height: 24px;
}
```

CSS comments

```css
/* this is a comment */
.intro {
  /* comments can contain text or “commented out” code
  color: #f0f0f0; */
  background-color: rgb(100, 0, 200);
}
```

`width` and `height`

Set the width and/or height of an element. Values are a percent or a number followed by a length unit, like 100% or 200px.

Code examples:

```
img {
  width: 100%;
}

div {
  height: 200px;
}
```

`overflow`

`overflow-x` (horizontal)

`overflow-y` (vertical)

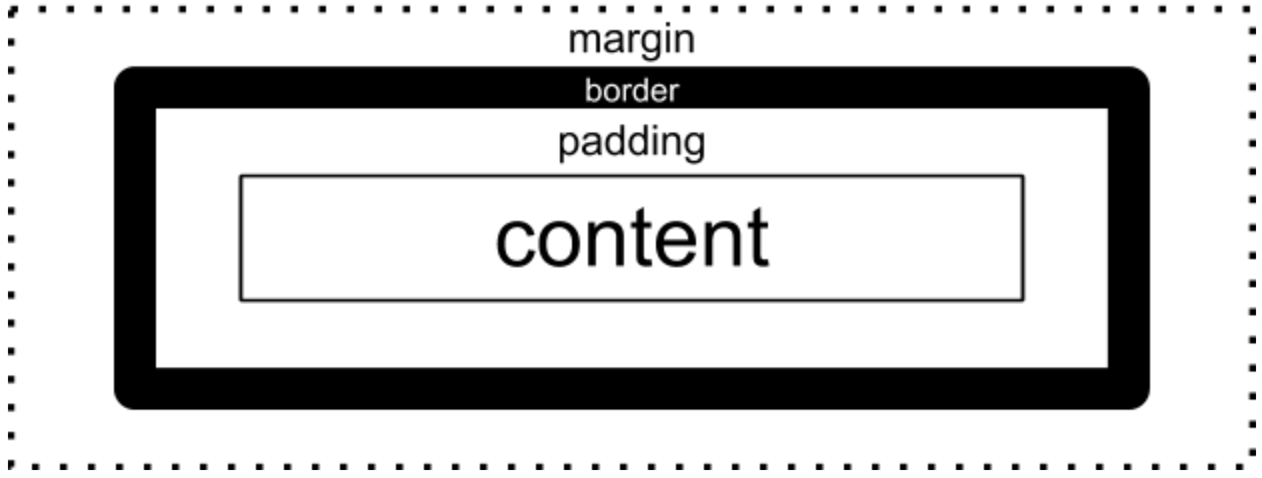
When content overflows its box, specify whether it should be hidden or to add scrollbars when an element's content is too big to fit. If setting overflow to anything other than it’s default value of `visible`, use `auto` to only show the scrollbars when the content does actually overflow, and only set overflow in the direction where you expect a user to scroll. Avoid scrollbars within scrollbars.

visible	default, content may flow outside its box
hidden	content is clipped, no scrolling
scroll	content is clipped, scrollbars always show
auto	clipped with scrollbars only if it overflows

Code examples:

```
p {
  height: 200px;
  overflow-y: auto;
}
```

CSS box model



`margin`

`margin-top margin-right margin-bottom margin-left`

Margin is the transparent area around the box that separates the box from other elements. Can also be applied individually to the top, left, bottom, or right margins.

`border` (shorthand)

`border-width border-style border-color`
`border-top border-right border-bottom border-left`

Sets the border width, style, and color. Can also be applied individually to `border-style`, `border-width`, `border-color`, or to the top, left, bottom, or right borders. Border style options are `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, or `outset`.

`padding`

`padding-top padding-right padding-bottom padding-left`

Padding is the space between the border and the content. Can also be applied individually to the top, left, bottom, or right padding.

Code example:

```
div {
  margin: 40px;
  padding: 20px;
  border: 5px solid red;
}
```

background-image

Sets a background image for an element. It's a good idea to always set a `background-color` as well to be used in case the image is unavailable. By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally. Set the `background-position`, `background-size`, and `background-repeat` to tile, stretch, or fit the background image to the element as desired.

background-position

Sets the starting position of a background image.
`xpos` can be `left`, `right`, `center`, or a CSS unit like `50%` or `20px`
`ypos` can be `top`, `bottom`, `center`, or a CSS unit like `50%` or `20px`

background-repeat

Sets if/how a background image will be repeated.

`background-repeat: repeat;`	(default) image is repeated both ways
`background-repeat: repeat-x;`	image is repeated horizontally
`background-repeat: repeat-y;`	image is repeated vertically
`background-repeat: no-repeat;`	image is not repeated

background-size

Specifies the size of the background images using keywords `auto`, `cover`, or `contain` or with CSS width and height units like `50%` or `200px`

`background-size: auto;`	(default) image is shown in its original size
`background-size: cover;`	resized, stretched, clipped to cover the container
`background-size: contain;`	resized to fit the container but not stretched to fill it
`background-size: 50% 50%;`	resized to 50% of the width and height
`background-size: 50px;`	width is 50px and height is auto

Code example:

```css
body {
  background-image: url('image.gif');
  background-color: #f0f0f0;
  background-position: center;
  background-repeat: no-repeat;
  background-size: contain;
}
```

position

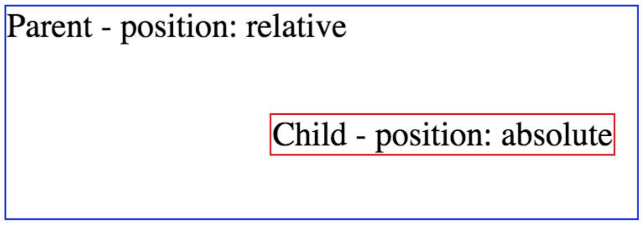
Specifies whether elements should be positioned in the regular document flow or positioned relative to another element or the browser window.

`position: static;`	● (default) normal position in order of document flow ● won’t be an anchor point for positioned child elements
`position: relative;`	● place element relative to normal position ● mostly used to anchor child absolute positioned elements
`position: absolute;`	● place relative to page or to the first positioned parent ● put inside relative or absolute positioned parent element
`position: fixed;`	● place element relative to the browser window ● doesn’t move when the page is scrolled

Code example:

```
#parent {
  position: relative;
  border: 1px solid blue;
  width: 300px;
  height: 100px;
}
#child {
  position: absolute;
  border: 1px solid red;
  top: 50%;
  right: 10px;
}
```

```
<div id="parent">
  Parent - position: relative
  <div id="child">
    Child - position: absolute
  </div>
</div>
```



z-index

Specifies the stack order of an element. Only works on positioned elements. Elements with a higher stack order are placed in front of elements with a lower stack order.

Code example:

```
.menu {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: 5;
}
```

CSS Flexbox

`display: flex`

Defines a flex container and enables a flex context for all its direct children or “flex items.”

`flex-direction`

Defines the direction flex items are placed in the flex container. Aside from optional wrapping, Flexbox is a tool for single-direction layouts. Flex items are laid out either in horizontal rows or vertical columns.

`flex-direction: row;`	(default) items laid out horizontally left to right
`flex-direction: row-reverse;`	flex items laid out horizontally right to left
`flex-direction: column;`	flex items laid out vertically top to bottom
`flex-direction: column-reverse;`	flex items laid out vertically bottom to top

`flex-wrap`

Flex items try to fit in one line by default, but items can wrap to multiple lines with this property.

`flex-wrap: nowrap;`	(default) flex items will be in one line
`flex-wrap: wrap;`	flex items can wrap to multiple lines top to bottom
`flex-wrap: wrap-reverse;`	flex items can wrap to multiple lines bottom to top

`justify-content`

Defines how items are laid out along the main axis, which by default is the horizontal axis when the `flex-direction` is row.

`justify-content: flex-start;`	(default) flex items are packed towards the start
`justify-content: flex-end;`	flex items are packed towards the end
`justify-content: center;`	center flex items in the main axis
`justify-content: space-between;`	distributed evenly from start to end
`justify-content: space-around;`	distributed with space at the start and end
`justify-content: space-evenly;`	like `space-around` but with equal space at the start and end

`align-items`

Defines how items are laid out across the cross axis, which by default is the vertical axis.

`align-items: flex-start;`	(default) aligned to start (top by default) of container
`align-items: flex-end;`	flex items aligned to end (bottom by default) or container
`align-items: center;`	center flex items in the cross axis (vertical by default)
`align-items: baseline;`	align flex items by their content baseline
`align-items: stretch;`	stretch flex items to fill the container