CSS

2022

Link

```
<link rel="stylesheet" href="…css">
```

Selectors/Combinators

Global selector:

```
*{ }
```

Selects both the input and button:

```
input, button { }
```

Selects a child h2 inside a tag with id='myid':

```
#myid h2 {  }
```

Selects only p elements with class center (no blank space between p and center):

```
p.center {  }
```

Selects all selectors and applies the same properties to each one of them:

```
h1, h2, p {   }
```

To select the classes of the <div class="popup opened"></div>

```
.popup.opened {  }
```

To select the child div with class="opened"

<div class="popup">

        <div class="opened">

```
.popup .opened {  }
```

blank space

To select all <p> inside a <div>:

```
div p {   }
```

To select all <p> which are children[1] of a <div>:

```
div > p {   }
```

To select all <p> appeared *immediately* after a <div>:

```
div + p {   }
```

[1] direct descendant

To select all immediate relatives[2] <p> of <div>:

```
div ~ p {   }
```

Pseudo-classes

```
:first-child
```

For example,

```
p  i:first-chile {   }
```

selects the first of <i>s that belongs to <p>.

```
:focus
```

```
:nth-of-type(2 / odd / even / βn+a)
```

For example,

```
p: nth-of-type(2)
```

selects the second <p>.

Pseudo-elements

```
p::first-line {  }
```

selects the first line of <p>

```
p::first-letter {  }
```

selects the first letter of <p>

```
::before
```

inserts some content before the element. For example,

```
h1::before {content: url('smiley.gif')};
```

```
::after
```

```
::marker
```

Marker selects the bullets of a list.

```
::selection
```

Selects the region selected by the user.

[2] all <p> appeared continuously after the <div>

Attribute Selectors

`a[target] {   }`

selects all <a> with target attribute.

`a[target="_blank"] {   }`

selects all <a> with target="_blank" attribute.

`a[target ~="value"] {    }`

selects all <a> with target containg the word "value".

`|=` means "begins with" and selects whole words only.

`^=` means "begins with".

`$=` means "ends with".

`*=` means "contains".

Units

`mm, cm, px, pt, em, vw, vh, rem, %, vmin, vmax`

`1em` = size like the font = 16px (default font size)

`2em` = size double of the font

`rem` = em of the root

`vmin` = 1% of viewport's smallest dimension

Responsive units

`1vw` = 1% of the viewport's width

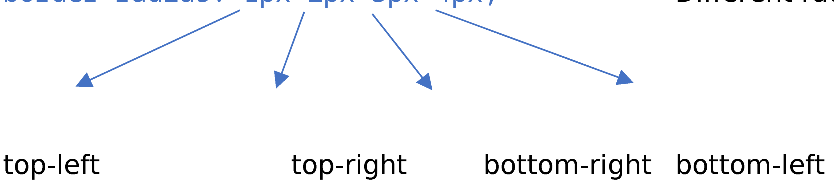
`1vh` = 1% of the viewport's height

Border

`border: 5px solid red;`

`border-radius: 5px;` Same radius on all sides

`border-radius: 1px 2px 3px 4px;` Different radius on sides



Background

```
background-image: url( );

background-size:  auto;

                  100px 50px;

                  50% 70%;

                  cover;

                  contain;
```

To repeat the background on the x-axis: `background-repeat: repeat-x;`

To make the background scroll: `background-attachment: scroll;`

To keep the background fixed: `background-attachment: fixed`;

Shadow

```
text-shadow: 1px 2px 5px red;
```

horizontal distance   vertical distance   spread

For a shadow around a box:

```
box-shadow: inset 1px 2px 10px black;
```

inside shadow   x-distance   y-distance   spread

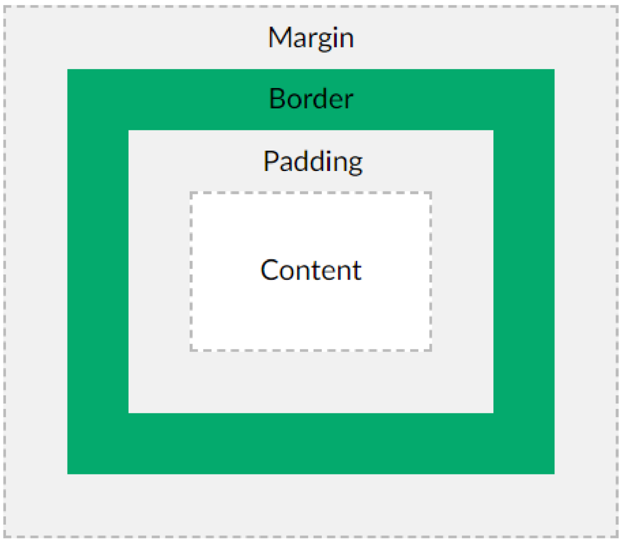
Positioning

```
static; (default)

relative; (to static)

fixed;

absolute;  (position relative to the ancestor)

sticky;
```

To draw over other object, we use

```
z-index
```

Margins



To inherit the margin of the parent:

```
margin-left: inherit;
```

To center the element:

```
margin: auto;
```

Padding

To inherit the padding of the parent:

```
padding-left: inherit;
```

Padding increases the width of the element. To avoid this, use:

```
box-sizing: border-box;
```

Outline

Same as border without offset.

Offset

Empty space between the outline and the border.

Text

Some properties of text are:

- direction
- vertical-align
- text-decoration: overline / line-through / underline / none
- text-transform: uppercase / lowercase / capitalize

Center text

Center text horizontically:

```
text-align: center;
```

Center text vertically:

```
height:100px;
```

```
line-height:100px;
```

Related to text properties

`white-space: wrap / nowrap;`

`text-indent`

`letter-spacing`

`line-height`

`word-spacing`

In a text area, to avoid a grab at the bottom-right side (which is used for resizing), we write:

`resize:none;`

Link

`a:link`

`a:visited`

`a:hover`

`a:active`

List

`list-style-type`

`list-style-position`

`list-style-image`

Table

`border-collapse:collapse;`

`border-bottom`

`border-top`

`tr:hover`

`tr:nth-child(even)` which selects the even rows.

Responsive table

Put the table inside a <div> with overflow-x:auto.

Display

```
display:    inline-block;

            block;

            inline;

            none;

            flex;
```

`display: none` hides the element but it takes some space. On the other hand,

`visibility:hidden` hides the element without keeping the empty space.

Width/Height

`max-width`

defines the maximum width of the element, even if the current width of the element is less than that.

Overflow

```
overflow:   visible;

            hidden;

            scroll;

            auto;
```

`overflow-x`

`overflow-y`

Float

```
float:      left;

      right;

      none;

      inherit;

clear:      none;

      left;

      right;

      both;

      inherit;
```

clearfix

Old version of clearfix is:

```
.clearfix {

            overflow: auto;

}
```

New version of clearfix is:

```
.clearfix ::after {

                  content:"";

                  clear: both;

                  display:table;

}
```

Flex

```
flex-container {

                  display:flex;

                  flex-wrap:nowrap;

}
```

Properties of the container

```
flex-direction: column / column-reverse / row / row-reverse

flex-wrap: wrap / nowrap / wrap-reverse
```

`flex-flow: row wrap;`   shortens the previous expressions

Aligh horizontically:

`justify-content: center / flex-start / flex-end / space-around / space-between`

Align vertically:

`align-items: center / flex-start / flex-end / stretch (default) / baseline`

Aligh the lines:

`align-content: space-between / space-around / stretch / center / flex-start / flex-end`

Properties of the flex child items

`order:3;`	the current child item will occupy the position 3
`flex-grow:8;`	the current child item will be 8 times larger than the others
`flex-shrink:2;`	the current child item will become 2 times smaller than others
`flex-basis:100px;`	the default length of an item
`align-self`	same as align-items but only for the current flex child.

$$\begin{tikzcd}
\texttt{flex:} & \texttt{0} \arrow[d] & \texttt{0} \arrow[d] & \texttt{200px;} \arrow[d] \\
& \text{flex-grow} & \text{flex-shrink} & \text{flex-basis}
\end{tikzcd}$$

`flex: 25%;` means `flex-basis:25%`

Button

`cursor:pointer`

Counters

`counter-reset: variable;`	means variable = 0
`counter-increment: variable;`	means variable++
`content: “my text” counter(variable)`	returns “my text” if variable = 2

Specificity Hierarchy

1000:	style attributes (inline)
100:	id
10:	attributes, classes, pseudo-classes
1:	element name, pseudo-element

Important

```
!important
```

We use !important to override other css settings, or to declare a style we do not want to change.

Transform 2D

```
transform:  translate(x, y);

            rotate(20deg);

            scale(m, n);    m times the width & n times the height

            scaleX(m);

            scaleY(n);

            skew(20deg);

            skewX(20deg);

            skewY(20deg);

            matrix(a,b,c,d,e,f)
```

where

- a = scaleX
- b = skewY
- c = skewX
- d = scaleY
- e = translateX
- f = translateY

Transition

```
transition: width 2s;
```

the property we change the duration of the change

The transition is executed after the property has changed.

```
transition: width 2s, height 3s, transform 2s;
```

```
transition-timing-function:   ease;   (default)

                              linear;

                              easy-in;

                              easy-out;

                              easy-in-out;
```

```
transition-delay:2s;

transition-duration:3s;

transition: width  2s  linear  1s;
```

property   duration   way   delay

```
transition: all 0.3s;
```

Media Queries

```
@media screen and (min-width:480px) {   }

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

@media screen and (max-width:992px) {   }

@media only screen and (orientation:landscape) {   }
```

Animation

```
@keyframes animationName {

      from {   } ←――――― from this style

      to {   } ←――――― to this style

}


@ keyframes animationName {

      0% {  } ←――――――――――――――――― beginning of the animation

      25% {   }

      50% {   }

      100% {   } ←――――――――――――――― end of the animation

}


animation-name: animationNama;

animation-duration: 4s;

animation-delay: 2s;      delay before the beginning of the animation

animation-iteration-count: infinite;

animation-iteration-count: 3;
```

`animation-direction:`	`normal;`	
	`reverse;`	
	`alternate;`	first normal, then reverse
	`alternate-reverse;`	first reverse, then normal

`animation-timing-function:`	`ease;`	(default)
	`linear;`	
	`easy-in;`	
	`easy-out;`	
	`easy-in-out;`	
	`cubic-bezier(n,n,n,n);`	

`animation-fill-mode:`	`none;`	default
	`forwards;`	element retains last keyframe style
	`backwards;`	element retains first keyframe style
	`both;`	both above

what style the element will have after the animation

```
animation: name duration function delay iteration direction;
```

For example,

```
animation: example 5s linear 2s infinite alternate;
```

Smooth Scrolling

```
html {

      scroll-behavior:smooth;

}
```