CSS CheatSheet Books

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML

▼ **Table of Contents**

Inline CSS

`style` attribute is used to define CSS properties at each HTML element.

```
<h1 style = "color:blue; font-size:40px; font-style: italic;"> One Compiler </h1>
```

Internal CSS

You can define CSS properties using the `<style>` tag in the `<head>` section.

```
<head>
    <style>
        body {background-color: pink;}
        h1   {color: red;}
        h2   {color: green; font-size : 40px; font-style: italic;}
    </style>
</head>
```

External CSS

`<link>` tag is used to refer to an external CSS file.

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

CSS Selectors

You can select elements based on their name

```
    div {
        font-familt:'Inter',sans-serrif;
        max-width:400px;
    }
```

or you can use both class based or id based css selection.

```
    // classed based
    .container {
        background:red;
        height:600px;
    }
    // id based
    #container {
        background:purple;
        margin:10px;
    }
```

there are also more fun selectors for different use cases that could be helpful in the long run when you're having a massive project and making tons of classes won't be viable to handle, such as:

Selector	What it does?
*	* selector selects and gives you all the elements individually, so you're essentially selecting all the elements one by one rather than

Selector	What it does?
	having a parent controlling the styling
element1,element2	The example of this syntax could be `div,p{ ... }` this allows you to select all element1 and element2 from the html
element1 element2	Replacing the comma(,) with a space helps you in selecting all the element2 which are inside element1, for eg: `div a{ ... }` means all the anchor tags(a) which are inside a div
element1 > element2	Much like how the space works, it gives the all the element2 who are directly inside element1 for eg.: `div>p{ ... }` means all the `<div><p> .... </p></div>` will be selected but not `<div><span><p> </p></span></div>` because p is not the direct child
element1 + element2	Quite rarely used but quite useful in some cases, it selects element2 which is directly after element1 for eg.: `div+p{ ... }` means all the `<div> ... </div><p> ... </p>` will be selected
element1 ~ element2	Also quite rarely used but useful in some cases, it selects the same as the + selector but rather than what + does by selecting only a single element, it selects all the elements following

```
* {
    background:red;
    min-height:100vh;
}
div,p {
    background:purple;
}
div p {
    background:yellow;
}
div > p {
    background:green;
}
div + p {
    background:blue;
}
div ~ p {
    background:white;
}
```

Units

Units are used for interpreting length in your css code value. These are used in properties like `width` , `height` , `font-size` , `margin` , `padding` etc.

- **Absolute units**

Units	description
in	used for inches
px	used for absolute pixels (usually 1/96th of an inch)
pt	points, usually 1/72th of an inch
pc	picas, usally 12 points
cm	used for centimeters
mm	used for millimeters
Q	used for Quarter-millimeters, (Usually 1/40th of 1cm)

- **Relative units(preferred)**

Units	description
%	relative to the size of the parent for eg. `100%` means filling all inside the parent
em	relative to the font size of the element
rem	relative to the font size of the root element
vw	relative to the viewport's width, for eg.: `2vw` would be `2% of the width` of the viewport
vh	relative to the viewport's height, for eg.: `6vh` would be `6% of the height` of the viewport

Units	description
vmax	relative to the viewport's bigger dimension which could be either the height or the width, for eg.: `3vmax` would mean `3vh` if the height is more than the width and would mean `3vw` otherwise
vmin	similar to vmax, differs because it would consider the smaller dimension of either the height or the width
ch	it is relative to the width of zero (0)
ex	it is relative to the x height of the current font

> relative units are much more preferred nowadays as their are just too many devices with varying screen heights, widths, pixel densities etc.

```
    * {
        background: red;
        min-height: 100vh;    //relative unit
    }
    div,p {
        background: purple;
        max-width: 200px      //absolute unit
    }
```

Display Properites

The display CSS property sets whether an element is treated as a block or inline element and the layout used for its children, such as flow layout, grid or flex. Formally, the display property sets an element's inner and outer display types.

```
.container {
        // various display values
        display: block;
        display: inline;
        display: inline-block;
        display: flex;
        display: inline-flex;
        display: grid;
        display: inline-grid;
        display: flow-root;
        display: table;
        display: list-item
}
```

FlexBox

You can use Flexbox to manage alignment and position of your elements.

To use Flexbox, give this property to the parent element:

```
.parent {
    display:flex;
}
```

To align the elements towards the main axis (by default it's horizontal), we use `justify-content` .

Vlaues	description
flex-start	Items are packed towards the start
center	Items are packed on the center
flex-end	Items are packed towards the end
space-around	Items are equally distributed with equal space aroun them
space-between	Items are evenly distributed .first item at the start and last items at the end
space-evenly	Items are evenly spaced with same amount space between them

To align the elements towards the cross-axis, we use `align-items` .

Vlaues	description
flex-start	Items are packed towards the start of cross axis
center	Items are packed on the center of cross axis
flex-end	Items are packed towards the end of the cross axis

By default, the flex direction is set to row (horizontal). To switch the flex direction to column (vertical), use:

The direction of flex is consider the main axis and the other axis consider as cross axis

```
.parent {
    display:flex;
    flex-direction:column;
}
```

CSS Grid

CSS grid is another way to properly align your HTML elements.

to create a new grid use

```
.box {
    display:grid;
}
```

CSS grid is made of two things: columns and rows. Using `grid-template-rows` and `grid-template-columns`, you can define how many rows and columns you want.

```
.box {
    display:grid;
    grid-template-columns:400px 300px 200px;
    grid-template-rows:50px 70px  60px;
}
```

You can use grid with a special unit called `Fr (fraction)`, which refers to a portion of remaining space.

```
.box {
    display:grid;
    grid-template-columns:1fr 1fr 1fr;
    // or
    grid-template-columns: repeat(3,1fr)
}
```

Variables

Variables are a great way to make your css more manageable, so you're not editing the values you want to be consistent on multiple instances of its usage. It promotes consistency and overall management of the code.

```
:root{
  --primary-color: #ffffff;
}
body{
  background-color: var(--primary-color);
}
```

Animations

CSS animations allow one to animate transitions or other media files on the web page.

Property	Description	Example
Animation	A shorthand property for setting all the animation properties	`animation: example 5s linear 2s infinite alternate;`
Animation-name	Specifies the name of the @keyframes animation	`animation-name: myanimation;`
Animation-duration	Specifies how long time an animation should take to complete one cycle	`animation-duration: 10s;`
Animation-timing-function	Specifies the speed curve of the animation	`animation-timing-function: ease;>`
Animation-delay	Specifies a delay for the start of an animation	`animation-delay: 5ms;`
Animation-iteration-count	Specifies the number of times an animation should be played	`animation-iteration-count: 3;`
Animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles	`animation-direction: normal;`

Property	Description	Example
Animation-play-state	Specifies whether the animation is running or paused	`animation-play-state: running;`
Animation-fill-mode	Specifies whether the animation is running or paused	`animation-fill-mode: both;`

Transitions

Transitions let you define the transition between two states of an element.

Property	Description	Example
Transition	A shorthand property for setting the four transition properties into a single property	`transition: width 2s linear 1s;`
Transition-property	Specifies the name of the CSS property the transition effect is for	`transition-property: none;`
Transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete	`transition-duration: 2s;`
Transition-timing-function	Specifies the speed curve of the transition effect	`transition-timing-function: ease-in-out;`
Transition-delay	Specifies a delay (in seconds) for the transition effect	`transition-delay: 20ms;`

Media queries

CSS media queries empowers you greatly when you’re creating and developing sites that are reponsive i.e. look and function well on different screen sizes and pixel densities.

When using media queries we can adopt the following approaches

- **Desktop first approach**

```
@media all and (min-width: 1024px) and (max-width: 1280px) {
    /* Targets desktop screens */
}

@media all and (min-width: 768px) and (max-width: 1024px) {
    /* Targets tablet landscape */
}

@media all and (min-width: 480px) and (max-width: 768px) {
    /* Targets tablet portrait */
}

@media all and (max-width: 480px) {
    /* Targets mobile screens*/
}
```

- **Mobile first approach**

```
@media only screen {
    /* Targets mobile screens with width < 641pz */
}

@media only screen and (min-width: 641px) {
    /* Targets tablet screens with width > 641px */
}

@media only screen and (min-width: 1025px) {
    /* Targets large screens(desktop) with width > 1025px */
}

@media only screen and (min-width: 1441px) {
    /* Targets xlarge screens with width > 1441px */
}

@media only screen and (min-width: 1921px) {
    /* Targets xxlarge screens with width > 1921px */
}
```

- **Orientation first approach**

```
@media screen and (orientation:portrait) {
    /* Add portrait styles here */
}
@media screen and (orientation:landscape) {
    /* Add landscape styles here */
}
```

Pseudo - Class

A pseudo-class is used to define a special state of an element.

- For example :

```
 a:link {
  color: #FF0000;
}

a:visited {
  color: #00FF00;
}

a:hover {
  color: #FF00FF;
}

a:active {
  color: #0000FF;
}
```

Property	Description
`:active`	an activated element
`:focus`	an element while the element has focus
`:visited`	a visted link
`:hover`	an element when you mouse over it
`:link`	an unvisited link
`:disabled`	an element while the element is disabled
`:enabled`	an element while the element is enabled
`:nth-child(n)`	an element that is the n-th sibling
`:nth-last-child(n)`	an element that is the n-th sibling counting from the last sibling