COMMON CSS PROPERTIES

CSS Comment	`/* comment */`
CSS Rule Syntax	`selector { property: value; }`
Example: Targets all <p> tags to change text color to red	`p { color: red; }`
FONT PROPERTIES	
Specify font for an element	`font-family: "Times New Roman", Times, serif;`

Set font size for an element	`font-size: 12px;`
	`font-size: [large|small];`
	`font-size: 200%;`

Set font to normal or italic	`font-style: italic;` `font-style: normal;`
Set font to normal or bold	`font-weight: bold;` `font-weight: normal;`
Set to small-caps or normal	`font-variant: small-caps;` `font-variant: normal;`
Shorthand	`font: style, variant, weight, size/height, family`
TEXT PROPERTIES	
Set the color of text	`color: red;`
Set space between characters	`letter-spacing: normal;`
Set space between lines	`line-height: 1.6px;` `line-height: 80%;`

Specify horizontal alignment of text	`text-align: center;`
	`text-align: right;`
	`text-align: left;`
	`text-align: justified;`

Specify decoration applied to text elements	`text-decoration: none;`
	`text-decoration: overline;`
	`text-decoration: underline;`
	`text-decoration: overline underline;`
	`text-decoration: line-through;`

Set text to uppercase, lowercase or capitalize	`text-transform: uppercase;`
	`text-transform: lowercase;`
	`text-transform: capitalize;`

Set how text overflow displays	`text-overflow: clip;`
	`text-overflow: hidden;`
	`text-overflow: ellipses;`

BACKGROUND PROPERTIES	
Set the background color	`background-color: red;`
Set image as a background	`background-image: url("path/to/img.jpg");`
Set starting position of background image	`background-position: [center|right|left];` `background-position: [x% y%][x_pos y_pos];`
Set size of background	`background-size: auto|width height|cover|contain`
Set repeat of image	`background-repeat: no-repeat|repeat|repeat-y|repeat-x`
Background shorthand	`background: color, image, position, size, repeat, origin, clip, attachment`
BORDER PROPERTIES	
Set width for border lines	`border-width: [medium|thin|thick|px]`
Set style of border lines	`border-style: none|hidden|dotted|dashed|double|single|groove|ridge|inset|outset|initial|inherit`
Set color of border	`border-color: red;`
Set radius of border	`border-radius: 50px;`
Border shorthand property	`border: width, style, color`
SPACING PROPERTIES	

Padding: space between content and border
(Same values for margin)
	`padding: all-sides`
	`padding: top-bottom, right-left`
	`padding: top, right-left, bottom`
	`padding: top, right, bottom, left`

Space between elements	`margin: top, right, bottom, left`
SIZING PROPERTIES	
Set height of an element	`height: [auto|initial|inherit|px|% of parent]`
Set width of an element	`width: [auto|initial|inherit|px|% of parent]`

Set maximum width/height
 	
Set minimum width/height:
	`max-width: [none|length|initial|inherit];`
	`max-height: [none|length|initial|inherit];`
	`min-height: [none|length|initial|inherit];`
	`min-width: [none|length|initial|inherit];`

LIST PROPERTIES	
Set style of list marker	`list-style-type: [circle|square|initial|inherit]`
Set position of list markers	`list-style-position: [outside|inside|initial]`
Set image as list marker	`list-style-image: url('/path/to/img.jpg');`
Shorthand for list	`list-style: type, position, image`

CSS COLORS

Specify using color name	`color: red;`
	`color: darkgrey;`
	`color: MediumSeaGreen;`

Specify using rgb:
Example:
	`rgb(red, green, blue)`
	`color: rgb(255,0,255)`
	`color: rgb(50, 100, 150)`
	`rgb(90, 90, 90)`

Specify using rgba:
A = Alpha:
0: fully transparent
1: fully opaque
	`rgba(red, green, blue, alpha)`
	
	`color: rgba(255, 0, 0, 0.3);`
	`color: rgba(255, 0, 0, 1);`

Specify using hexadecimal
Example:
	`#rrggbb`
	`color: 009900;`
	`color: ff22aa;`

Three-digit hex code:
Equivalent to ff0000:
Equivalent to ffcc99:
	`#rgb`
	`color: f00;`
	`color: #fc9;`

Specify using hsl
Hue = degree from 0 – 360
0: red, 120: green, 240: blue
Saturation = 0 – 100%
0 = grey, 100 = full color
Lightness = 0 – 100%
0 = black, 100% = white
	`hsl(hue, saturation, lightness)`
	
	`color: hsl(0, 100%, 50%);`
	`color: hsl(100, 50%, 25%);`
	`color: hsl(240, 75%, 50%);`
	`color: hsl(360, 50%, 60%);`
	`color: hsl(300, 20%, 50%);`

CSS UNITS OF MEASUREMENT

Relative in CSS2, absolute in CSS3 (1/96th of an inch)	`px`
Value of the font-size property of the element it is used on.	`em`
Value relative to the parent element	`%`
Value relative to the x-height of the current font	`ex`
Value relative to width of "0" (zero) of the current font	`ch`
Value relative to font-size of the root element	`rem`
Value from 0 – 100, relative to 1% of the width/height of viewport	`vw/vh`
Absolute value of an inch, centimeter and millimeter	`in/cm/mm`
Absolute value of a point (pt = 1/72 of an inch) & pica (pc = ⅙ of an inch)	`pt/pc`

SELECTORS

Universal (all) selector	`* { ... }`
Element selector: Selects all p tags	`p { ... }`
Grouped selector: Selects all p, div, and h1 tags	`p, div, h1 { ... }`
Class selector: Selects all elements with class name 'cl1'	`.cl1 { ... }`
Selects only p elements with the class name 'c1'	`p.cl1 { ... }`
Id selector: Selects element with id name 'id1'	`#id1 { ... }`
Selects only a div element with the id name 'id1'	`div#id1 { ... }`
Attribute selector: Selects elements with attribute	`[attribute]`
Example: Targets elements with href attribute	`HTML: <a href="#">Link</a>` `CSS: [href] { ... }`
Select specific tag with given attribute	`tag[attribute]`
Example: Matches img tags with alt attribute	`img[alt]`
Matches attributes set to value	`[attribute="value"]`
~=: Matches attribute value containing 'example'	`[attribute~="example"]`
^=: Matches attribute value beginning with 'example'	`[attribute^="test"]`
$=: Matches attributes value ending with 'example'	`[attribute$="test"]`

combinators

Descendant: All elements contained within another	`parent descendant { … }`
Example: Selects all <p> elements within elements	`div p { … }`
Child: Element that is a direct child of specified element	`parent>child { … }`
Example: Selects <p> elements that are directly inside div	`div>p { … }`
Adjacent Sibling: Targets elements directly after another	`sib1+sib2 { … }`
Example: Selects all <h2> tags directly after <h1> tags	`h2+h1 { … }`
General Sibling: Targets all elements after another	`sib1~sib2 { … }`
Example: Selects all <p> tags after <h2> tags	`h2~p { … }`

pseudo selectors

Syntax	`selector:pseudo-class {`
	`  property: value;`
	`}`

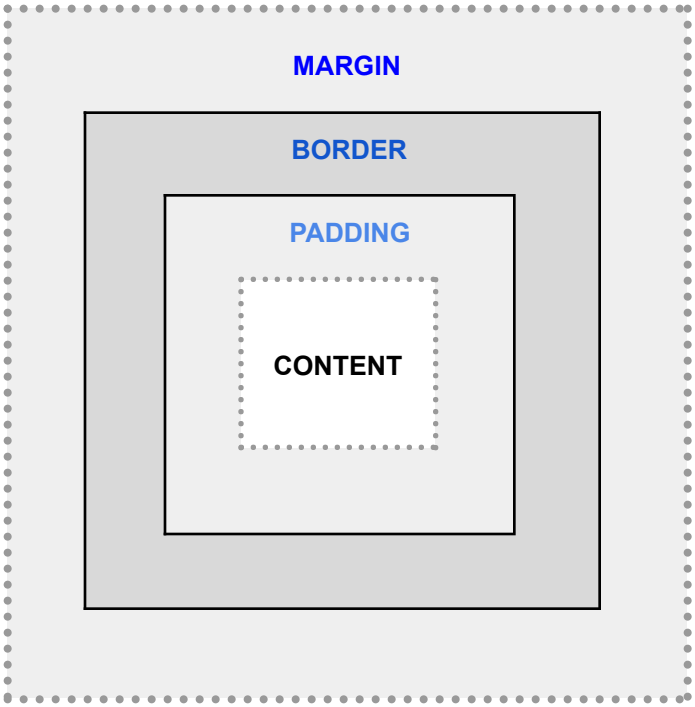
Hover: Targets elements with mouseover	`element:hover`
Example: Targets all p tags with the class 'target' when they are being hovered over	`p.target:hover { … }`
Example: Targets all p tags within a div when the div element is being hovered over	`div:hover p { … }`

<a> tag:
Unvisited link
Visited link
Mouse-over link
Selected (click-down) link
	`a:link`
	`a:visited`
	`a:hover  /* after link & visited */`
	`a:active  /* after hover */`

<input> tag:
Input elements that are checked
Input elements that are disabled
Input elements that are enabled (default)
Input elements that have focus (clicked on)
Input elements within a specific range
Input elements with an invalid value
Input elements with a valid value
Input elements with a required attribute
Input elements without a required attribute
	`input:checked`
	`input:disabled`
	`input:enabled`
	`input:focus`
	`input:in-range`
	`input:invalid`
	`input:valid`
	`input:required`
	`input:optional`

Elements that are the first child	`element:first-child`
Example: Matches first <i> inside a <p> tag	`p i:first-child`
Elements that are the last child	`element:last-child`
Elements that are the only child of its parent	`element:only-child`
Every child element whose index is odd	`element:nth-child(odd)`
Every child element whose index is even	`element:nth-child(even)`
Every <p> element that is the first <p> element of its parent	`p:first-of-type`
Every <p> element that is the last <p> element of its parent	`p:last-of-type`
Every <p> element that is the only <p> element of its parent	`p:only-of-type`

BOX MODEL



ELEMENT WIDTH:
LeftMargin
+ LeftBorder
+ LeftPadding
+ Width of Content
+ RightPadding
+ RightBorder
+ RightMargin

ELEMENT HEIGHT:
TopMargin
+ TopBorder
+ TopPadding
+ Height of Content
+ BottomPadding
+ BottomBorder
+ BottomMargin

COLLAPSE

margin-collapse

The margin between two elements will be the larger of the two margins, not the sum.

The resulting vertical space between h1 and h2 will be 100px, not 200px:

```
h1 {
  border: 1px solid black;
  margin: 100px;
}

h2 {
  border: 1px solid black;
  margin: 100px;
}
```

Heading 1

[100px]

Heading 2

border-collapse

The border in a table can be set to be separate or a single, collapsed border:

```
border-collapse: separate;    /* separate borders for cells and table */
border-collapse: collapse;    /* single border for cells and table */
```

INHERITANCE

UNIVERSAL PROPERTY VALUES	
Sets value to that of its parent element	`inherit`
Sets value to the initial value of that property	`initial`
Resets value to the browser's default style	`revert`
Resets value to one specified in previous cascade layer	`revert-layer`
Resets to natural value (inherited or initial)	`unset`

cascading rules & specificity

BROWSER	Browser default style	`<!-- basic style defined by browser -->`
USER STYLE	Overrides browser	`<!-- user-defined stylesheet by web user -->`
EXTERNAL	Overrides user style and previously defined stylesheets	`<link rel="stylesheet" href="style1.css" />` `<link rel="stylesheet" href="style2.css" />`

INTERNAL	Overrides External	`<style>`
		`  p { color: blue; }`
		`</style>`

INLINE	Overrides Internal	`<p style="color:red">`
SPECIFICITY		
Element	Most recently defined overrides others	`div { color: red; }` `div { color: blue; }`
Class/Pseudo Class	Overrides Element	`.container { color: green; }`
Id	Overrides Class	`#nav { color: yellow; }`

scoring specificity

A - B - C

– Add 1 to **A** for each **ID** in selector
– Add 1 to **B** for each **class** or pseudo-class in selector
– Add 1 to **C** for each **element** name
→ Read result as a 3-digit number

Example:

```
<div id="nav" class="links">
div { color: green; }          /* 0 0 1 */
div#nav { color: red; }        /* 1 0 1 */ -> div will be red
div.links { color: blue; }     /* 0 1 1 */
```

POSITION & DISPLAY

float: Used for positioning an element within its container	
The element floats to the left of its container	`float: left;`
The element floats to the right of its container	`float: right;`
The element does not float (default)	`float: none;`
The element inherits the float value of its parent	`float: inherit;`

Example: The image will float to the left of its container and other elements will be positioned to the right of it	`img {`
	`  float: left;`
	`}`

clear: Specifies position of the element that is next to a floating element.	
The element is not pushed below left or right (default)	`clear: none;`
The element is pushed below left floated elements	`clear: left;`
The element is pushed below right floated elements	`clear: right;`
The element is pushed below left & right floated elements	`clear: both;`
The element inherits the clear value from its parent	`clear: inherit;`

Example: The content of div1 will be floated to the left of its container. The content of div2 will be cleared below div1. Without this, the content of each div could overlap.	`.div1 {`
	`  float: left;`
	`}`
	`.div2 {`
	`  clear: left;`
	`}`

position	
top, right, bottom, left properties define offset (Example: element will be 10px offset from expected top position)	`top: 10px;` `right: 30px;`
Positioned according to the flow of the page (default)	`position: static;`
Positioned relative to its normal position	`position: relative;`
Positioned in a fixed location relative to viewport	`position: fixed;`
Positioned relative to the nearest positioned ancestor	`position: absolute;`
Positioned based on the user's scroll position.	`position: sticky;`

Example: div is positioned in its normal location, when the user scrolls down, it will become fixed at the top location (top: 0 means 0 pixels from the top of the viewport)	`div.sticky {`
	`  position: sticky;`
	`  top: 0;`
	`}`

z-index: Greater z-index of overlapped elements will appear on top	
Example: Without z-index, last defined element appears on top	`<div>Below</div>` `<div>In Front</div>`

z-index changes the order of which elements appear first	`.behind {`
	`  z-index: -1;`
	`}`
	`.middle {`
	`  z-index: 0;`
	`}`
	`.infront {`
	`  z-index: 1;`
	`}`

display	
Hide element (shows blank space in HTML page)	`visibility: hidden;`
Hide element (no blank space in HTML page)	`display: none;`
Change element to be displayed as block element	`display: block;`
Change element to be displayed as inline element	`display: inline;`

TRANSITIONS

Syntax	`transition: property duration timing-function delay|initial|inherit;`

Example:
Element with class 'change' will go from a width of 100 to 200 when hovered, with a transition time of 2 seconds
	`.change {`
	`  width: 100px;`
	`  transition: width 2s;`
	`}`
	`.change:hover {`
	`  width: 200px;`

Example:
Changes the div's height and width over 1 second, and color over 2 seconds, when hovered
	`div {`
	`  width: 100px;`
	`  height: 100px;`
	`  background: red;`
	`  transition: background 2s, width 1s, height 1s;`
	`}`
	`div:hover {`
	`  background: blue;`
	`  width: 120px;`
	`  height: 120px;`
	`}`

MEDIA QUERIES

Syntax	`@media mediatype and (mediafeature) {`
	`  CSS Rules {}`
	`}`

Example:
Changes text size when screen goes below 600px
	`@media screen and (max-width: 600px) {`
	`  body {`
	`  font-size: 12px;`
	`  }`
	`}`

Media types:
all: all media type devices
printer: printers
speech: screen readers
screen: computer screens, tablets, smart-phones etc.
	`@media all`
	`@media printer`
	`@media speech`
	`@media screen`

Media features:
Targets viewport less than num
Targets viewports greater than num
Targets screen orientation
	`max-width: [num]`
	`min-width: [num]`
	`orientation: [portrait|landscape]`

Multiple syntax:	`@media mediatype and (feature1) and (feature2)`
Example: When screen is between 600px and 900px	`@media screen and (max-width: 900px) and (min-width: 600px)`

FLEXBOX

Create flexbox container	`/* HTML */`
	`<div id="container">...</div>`
	`/* CSS */`
	`#container {`
	`  display: flex;`
	`}`

Create flex item
Note: An element must have a parent element with display: flex to become a flexible item
	`/* HTML */`
	`<div id="container">`
	`  <div>Flex Item1</div>`
	`  <div>Flex Item2</div>`
	`</div>`
	`/* CSS */`
	`#container > div {`
	`  background-color: grey;`
	`}`

flex container properties	

Change direction of flex items
row: Horizontally left to right (Default)
column: Vertically top to bottom
row-reverse: Horizontally right to left
column-reverse: Vertically bottom to top
	`flex-direction: row;`
	`flex-direction: column;`
	`flex-direction: row-reverse;`
	`flex-direction: column-reverse;`

Change wrap of flex items **wrap:** Wrap items when resizing window **nowrap:** Items do not wrap (Default)	`flex-wrap: wrap;` `flex-wrap: nowrap;`
Shorthand for setting direction and wrap Example: Stack in rows with wrap enabled	`flex-flow: direction wrap;` `flex-flow: row wrap;`

Set the horizontal alignment of flex items
center: center of container
flex-start: beginning of container (Default)
flex-end: end of container
space-around: space before, after, between
space-between: space between items
	`justify-content: center;`
	`justify-content: flex-start;`
	`justify-content: flex-end;`
	`justify-content: space-around;`
	`justify-content: space-between;`

Set the vertical alignment of items
center: the center of the container
flex-start: the top of the container
flex-end: the bottom of the container
stretch: Stretch items to fill container (Default)
baseline: align based on baseline of text
	`align-items: center`
	`align-items: flex-start;`
	`align-items: flex-end;`
	`align-items: stretch;`
	`align-items: baseline;`

Example: Flex items within the container will be centered vertically and horizontally	`#container {`
	`  display: flex;`
	`  justify-content: center;`
	`  align-items: center;`
	`}`

flex item properties	
Set the growth rate of a flex item flex item will grow at the same rate as others flex item will grow 2x wider than the others	`flex-grow: 1;` `flex-grow: 2;`

Set the shrink rate of a flex item
flex item will shrink at the same rate as others
flex item will shrink 2x more than the others
flex item will not shrink
	`flex-shrink: 1;`
	`flex-shrink: 2;`
	`flex-shrink: 0;`

Set the initial/minimum width of a flex item	`flex-basis: 200px;`

GRID

Create grid container
 	
grid: block-level; full width of container
inline-grid: inline-level; width of contents
	`/* HTML */`
	`<div id="container">...</div>`
	`/* CSS */`
	`#container {`
	`  display: [grid|inline-grid];`
	`}`

Define number of columns in grid and width Define number of rows in grid and height	`grid-template-columns: [auto|px|fr|%]` `grid-template-rows: [auto|px|fr|%]`

Example:
2 columns of equal width
3 rows: first is 100px, second is 25% of container, and last is 1 part of the available space
	`#container {`
	`  display: grid;`
	`  grid-template-columns: auto auto;`
	`  grid-template-rows: 100px 25% 1fr;`
	`}`

Using CSS function repeat() for column	`repeat(#_times, value_to_repeat);`
Example: Same as above, repeats auto 2x	`grid-template-columns: repeat(2, auto)`
minmax: at least x, but no greater than y	`repeat(#times, minmax(x, y))`
auto-fill: browser sets number of rows/cols	`repeat(auto-fill, value)`
auto-fit: remaining space occupied	`repeat(auto-fit, value)`

Set gap between rows (px, %, etc)
Set gap between columns (px, %, etc)
Shorthand for both
	`grid-row-gap: length;`
	`grid-column-gap: length;`
	`grid-gap: row-gap column-gap;`

Set auto placed items in new column/row	`grid-auto-flow: [column|row];`
Set size of auto-placed columns Set size of auto-placed rows	`grid-auto-column: size;` `grid-auto-row: size;`

Grid Area
 	
Griditem is given a grid-area of ex
 	
The grid item with **grid-area** set to 'ex' will span *2 columns* with 3 columns after
	`/* HTML */`
	`<div id="container">`
	`  <div id="griditem"></div>`
	`</div>`
	`/* CSS */`
	`#griditem {`
	`  grid-area: ex;`
	`}`
	`#container {`
	`  display: grid;`
	`  grid-template-areas: 'ex ex . . .';`
	`}`