SELECTORS Chapter

* CSS selectors are used to "find" (or select) the HTML elements you want to style.
* We can divide CSS selectors into five categories:
* Simple selectors (select elements based on name, id, class)
* [Combinator selectors](https://www.w3schools.com/css/css_combinators.asp) (select elements based on a specific relationship between them)
* [Pseudo-class selectors](https://www.w3schools.com/css/css_pseudo_classes.asp) (select elements based on a certain state)
* [Pseudo-elements selectors](https://www.w3schools.com/css/css_pseudo_elements.asp) (select and style a part of an element)
* [Attribute selectors](https://www.w3schools.com/css/css_attribute_selectors.asp) (select elements based on an attribute or attribute value)
* The CSS element Selector
* The element selector selects HTML elements based on the element name.
* Example:

p {  
  text-align: center;  
  color: red;  
}

* The CSS id Selector
* The id selector uses the id attribute of an HTML element to select a specific element.
* Example:

#para1 {  
  text-align: center;  
  color: red;  
}

* The CSS class Selector
* The class selector selects HTML elements with a specific class attribute.
* Example:

.center {  
  text-align: center;  
  color: red;  
}

* example only <p> elements with class="center" will be red and center-aligned:

p.center {  
  text-align: center;  
  color: red;  
}

* The CSS Universal Selector
* The universal selector (\*) selects all HTML elements on the page.
* Example:

\* {  
  text-align: center;  
  color: blue;  
}

* The CSS Grouping Selector
* The grouping selector selects all the HTML elements with the same style definitions.
* Example:

h1, h2, p {  
  text-align: center;  
  color: red;  
}

INSERT CSS Chapter

* Three ways to insert CSS
* External CSS
  + <head>  
     <link rel="stylesheet" href="mystyle.css">  
    </head>
* Internal CSS
  + <style>  
     h1 {  
      color: maroon;  
      margin-left: 40px;  
     }  
    </style>
* Inline CSS
  + <p style="color:red;">This is a paragraph.</p>
    - An inline style has the highest priority, and will override external and internal styles and browser defaults.

CSS BACKGROUND Chapter

* CSS background properties:
* background-color
  + The background-color property specifies the background color of an element.
  + Example: body {  
       background-color: lightblue;  
     }
  + The opacity property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent
  + Example: div {  
      background-color: green;  
      opacity: 0.3;  
     }
* background-image
  + The background-image property specifies an image to use as the background of an element.
  + Example: body {  
       background-image: url("paper.gif");  
     }
* background-repeat
  + By default, the background-image property repeats an image both horizontally and vertically.
* Example: body {  
     background-image: url("gradient\_bg.png");  
     background-repeat: repeat-x; - for horizontal //background-repeat: repeat-y; - for vertical  
   }
* Showing the background image only once is also specified by the background-repeat property.
* The background-position property is used to specify the position of the background image.
* Example: body {  
     background-image: url("img\_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;

}

* background-attachment
  + The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):
  + Example:   background-attachment: fixed;

background-attachment: scroll;

* background (shorthand property)
  + To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.
  + Example:

body {  
  background: #ffffff url("img\_tree.png") no-repeat right top;  
}

CSS BORDER STYLE Chapter

* The border-style property specifies what kind of border to display.

Example:

p {border-style: dotted;}

The following values are allowed:

* dotted - Defines a dotted border
* dashed - Defines a dashed border
* solid - Defines a solid border
* double - Defines a double border
* groove - Defines a 3D grooved border. The effect depends on the border-color value
* ridge - Defines a 3D ridged border. The effect depends on the border-color value
* inset - Defines a 3D inset border. The effect depends on the border-color value
* outset - Defines a 3D outset border. The effect depends on the border-color value
* none - Defines no border
* hidden - Defines a hidden border
* The border-width property specifies the width of the four borders.

Example:

p {  
  border-style: dotted;  
  border-width: thick;  
}

* The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.
* The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border)
* Example:

p.three {  
  border-style: solid;  
  border-width: 25px 10px 4px 35px; /\* 25px top, 10px right, 4px bottom and 35px left \*/  
}

* The border-color property is used to set the color of the four borders.
* p {  
    border-style: solid;  
    border-color: red green blue yellow; /\* red top, green right, blue bottom and yellow left \*/ **Can use one color tooo**}
* CSS Border Individual Sides
* From the examples on the previous pages, you have seen that it is possible to specify a different border for each side.
  + Example: p {  
      border-top-style: dotted;  
      border-right-style: solid;  
       border-bottom-style: dotted;  
       border-left-style: solid;  
     }
* Border Shorthand Property
* Example: border-width border-style(required) border-color

p {  
  border: 5px solid red;  
 }

* Border Radius
* The border-radius property is used to add rounded borders to an element:

CSS MARGIN Chapter

* The CSS margin properties are used to create space around elements, outside of any defined borders.

CSS has properties for specifying the margin for each side of an element:

* margin-top
* margin-right
* margin-bottom
* margin-left

All the margin properties can have the following values:

* auto - the browser calculates the margin
* *length* - specifies a margin in px, pt, cm, etc.
* *%* - specifies a margin in % of the width of the containing element
* inherit - specifies that the margin should be inherited from the parent element

Negative value are allowed.

* The margin property is a shorthand property for the following individual margin properties:

**Example:**

p {  
  margin: 25px 50px 75px 100px;  
}

* + top margin is 25px
  + right margin is 50px
  + bottom margin is 75px
  + left margin is 100px

p {  
  margin: 25px 50px 75px;  
}

* top margin is 25px
* right and left margins are 50px
* bottom margin is 75px

p {  
  margin: 25px 50px;  
}

* top and bottom margins are 25px
* right and left margins are 50px

p {  
  margin: 25px;  
}

* + all four margins are 25px
* You can set the margin property to auto to horizontally center the element within its container.

  margin: auto;

* Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins. (Only for Top and Bottom margin)

h1 {  
  margin: 0 0 50px 0;  
}  
  
h2 {  
  margin: 20px 0 0 0;  
}

CSS PADDING Chapter

* The CSS padding properties are used to generate space around an element's content, inside of any defined borders.
* The padding property is a shorthand property for the following individual padding properties:
* padding-top
* padding-right
* padding-bottom
* padding-left
* Everything is same as Margin property.
* The CSS width property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element ([the box model](https://www.w3schools.com/css/css_boxmodel.asp)). So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result.
* Here, the <div> element is given a width of 300px. However, the actual width of the <div> element will be 350px (300px + 25px of left padding + 25px of right padding):

div {  
  width: 300px;  
  padding: 25px;  
}

* To keep the width at 300px, no matter the amount of padding, you can use the box-sizing property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease.

div {  
  width: 300px;  
  padding: 25px;  
  box-sizing: border-box;  
}

OUTLINE Chapter

The outline-style property specifies the style of the outline, and can have one of the following values:

* dotted - Defines a dotted outline
* dashed - Defines a dashed outline
* solid - Defines a solid outline
* double - Defines a double outline
* groove - Defines a 3D grooved outline
* ridge - Defines a 3D ridged outline
* inset - Defines a 3D inset outline
* outset - Defines a 3D outset outline
* none - Defines no outline
* hidden - Defines a hidden outline

Everything is same as margin.

* The outline-width property specifies the width of the outline.
* The outline-color property is used to set the color of the outline.
* The outline property is a shorthand property for setting the following individual outline properties:
* outline-width
* outline-style (required)
* outline-color
* Example:

p.ex1 {  
  border: 1px solid black;  
  outline-style: solid;  
  outline-color: red;  
  outline-width: thin;  
}

CSS TEXT Chapter

* Property
* background-color: lightgrey;
* color: blue;
* text-align: center;

[ left / right / center / justify(every line have same width) ]

* The direction and unicode-bidi properties can be used to change the text direction of an element:

direction: rtl;  
  unicode-bidi: bidi-override;

* The vertical-align property sets the vertical alignment of an element.

vertical-align: baseline;

  vertical-align: text-top;

  vertical-align: text-bottom;

  vertical-align: sub;

  vertical-align: super;

* text-decoration: none; [ overline / line-through / underline]
* The text-transform property is used to specify uppercase and lowercase letters in a text.

text-transform: uppercase;  
  
  text-transform: lowercase;

  text-transform: capitalize;

* The text-indent property is used to specify the indentation of the first line of a text.
* letter-spacing: 3px;
* line-height: 0.8;
* word-spacing: 10px;
* white-space: nowrap;
* text-shadow: 2px 2px; ( horizontal shadow (2px) and the vertical shadow (2px))
* text-shadow: 2px 2px red;
* text-shadow: 2px 2px 5px red; (blur effect (5px))

FONT Chapter

* font-family: "Times New Roman", Times, serif;
* font-style: normal; [italic / oblique(similar to italic, but less supported)]
* font-weight: normal;   font-weight: bold;
* font-size: 100%;
* The font property is a shorthand property for:
* font-style
* font-variant
* font-weight
* font-size/line-height
* font-family

Example:

p {  
  font: italic small-caps bold 12px/30px Georgia, serif;  
}

CSS LINK Chapter

The four links states are:

* a:link - a normal, unvisited link
* a:visited - a link the user has visited
* a:hover - a link when the user mouses over it
* a:active - a link the moment it is clicked

CSS LIST Chapter

* The list-style-type property specifies the type of list item marker.
* list-style-type: circle; [square/upper-roman/lower-alpha/none]
* list-style-position: outside / inside;
* Shorthand ( list-style-type , list-style-position , list-style-image )

ul {  
  list-style: square inside url("sqpurple.gif");  
}

CSS TABLE Chapter

* table, th, td {  
    border: 1px solid black;  
  }
* border-collapse: collapse;
* vertical-align: bottom;
* border-bottom: 1px solid #ddd;
* tr:hover {background-color: #f5f5f5;} (Hover able Table)
* tr:nth-child(even) {background-color: #f2f2f2;} (Striped Table)
* Add a container element (like <div>) with overflow-x:auto around the <table> element to make it responsive.

CSS DISPLAY Chapter

* The display property specifies if/how an element is displayed.
* Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.
* One common use for display: inline-block is to display list items horizontally instead of vertically. The following example creates horizontal navigation links.
* Block Level Element
* A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).
* Examples of block-level elements:
* <div>
* <h1> - <h6>
* <p>
* <form>
* <header>
* <footer>
* <section>
* Inline Element
* An inline element does not start on a new line and only takes up as much width as necessary.
* Examples of inline elements:
  + <span>
  + <a>
  + <img>
* display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them.

Hiding an element can be done by setting the display property to none. The element will be hidden, and the page will be displayed as if the element is not there.

Display : Specifies how an element should be displayed.

* visibility: hidden; also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout.

Visibility : Specifies whether or not an element should be visible

* display: inline; display: block;

CSS POSITION Chapter

* The position property specifies the type of positioning method used for an element.
* Static
  + HTML elements are positioned static by default. position: static;
* Relative
  + An element with position: relative; is positioned relative to its normal position.
  + Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.
  + Example:

div.relative {  
  position: relative;  
  left: 30px;  
  border: 3px solid #73AD21;  
}

* Fixed (if you want element to always to be there)
  + An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
  + Example:

div.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 300px;  
  border: 3px solid #73AD21;  
}

* Absolute
  + An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
  + However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
  + Example: here, absolute div is within relative div

div.relative {  
  position: relative;  
  width: 400px;  
  height: 200px;  
  border: 3px solid #73AD21;  
}  
  
div.absolute {  
  position: absolute;  
  top: 80px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}

* Sticky
  + An element with position: sticky; is positioned based on the user's scroll position.
  + A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).
  + Example:

div.sticky {  
  position: -webkit-sticky; /\* Safari \*/  
  position: sticky;  
  top: 0;  
  background-color: green;  
  border: 2px solid #4CAF50;  
}

* Overlapping Element
* The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
* Example:

img {  
  position: absolute;  
  left: 0px;  
  top: 0px;  
  z-index: -1;  
}

CSS LAYOUT – OVERFLOW Chapter

* The CSS overflow property controls what happens to content that is too big to fit into an area. Example: overflow: visible;
* The overflow property has the following values:
  + visible - Default. The overflow is not clipped. The content renders outside the element's box
  + hidden - The overflow is clipped, and the rest of the content will be invisible
  + scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
  + auto - Similar to scroll, but it adds scrollbars only when necessary.
* The overflow-x and overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both):

Example:

div {  
  overflow-x: hidden; /\* Hide **horizontal** scrollbar \*/  
  overflow-y: scroll; /\* Add **vertical** scrollbar \*/  
}

CSS LAYOUT – FLOAT & CLEAR Chapter

* The float property is used for positioning and formatting content e.g. let an image float left to the text in a container. Example: float: right;

The float property can have one of the following values:

* left - The element floats to the left of its container
* right - The element floats to the right of its container
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent
* The clear property specifies what should happen with the element that is next to a floating element.

The clear property can have one of the following values:

* + none - The element is not pushed below left or right floated elements. This is default
  + left - The element is pushed below left floated elements
  + right - The element is pushed below right floated elements
  + both - The element is pushed below both left and right floated elements
  + inherit - The element inherits the clear value from its parent
* Box-sizing - Defines how the width and height of an element are calculated: should they include padding and borders, or not

CSS LAYOUT – HORIZONTAL / VERTICAL ALIGN

* To horizontally center a block element (like <div>), use margin: auto;
* To just center the text inside an element, use text-align: center;
* To center an image, set left and right margin to auto and make it into a block element.
* One method for aligning elements is to use position: absolute; Another method for aligning elements is to use the float property:
* If an element is taller than the element containing it, and it is floated, it will overflow outside of its container. You can use the "clearfix hack" to fix this (see example below).

.clearfix::after {  
  content: "";  
  clear: both;  
  display: table;  
}

<h2 style="clear:right">With New Modern Clearfix</h2>

<p>Add the clearfix hack to the containing element, to fix this problem:</p>

<div class="clearfix">

<img class="img2" src="pineapple.jpg" alt="Pineapple" width="170" height="170">

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...

</div>

* You can also use flexbox to center things Vertically.

Example:

.center {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  height: 200px;  
  border: 3px solid green;  
}

CSS COMBINATOR Chapter

* A combinator is something that explains the relationship between the selectors.

There are four different combinators in CSS:

* descendant selector (space)
  + The descendant selector matches all elements that are descendants of a specified element.

Example: div p {  
   background-color: yellow;  
 }

* child selector (>)
  + The child selector selects all elements that are the children of a specified element.
  + Example: div > p {  
       background-color: yellow;  
     }
* adjacent sibling selector (+)
  + The adjacent sibling selector is used to select an element that is immediately after another specific element.
  + Example: div + p {  
      background-color: yellow;  
     }
* general sibling selector (~)
  + The general sibling selector selects all elements that are next siblings (not immediate siblings) of a specified element.
  + Example: div ~ p {  
      background-color: yellow;  
     }

PSEUDO CLASSES Chapter

* To apply focus after clicking on input text box.

Example:

input:focus {

background-color: yellow;

}

PSEUDO ELEMENT Chapter

* ::after, ::before, ::first-line, ::first-letter

IMAGE OPACITY/TRANSPARENCY Chapter

* The opacity property specifies the opacity/transparency of an element.
* The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent:
* When using the opacity property to add transparency to the background of an element, all of its child elements inherit the same transparency. This can make the text inside a fully transparent element hard to read. If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:
* div {  
    background: rgba(76, 175, 80, 0.3) /\* Green background with 30% opacity \*/  
  }
* TEXT IN A TRANSPERENT BOX.

CSS BACKGROUND

CSS COLORS

CSS GRADIENTS

CSS SHADOW EFFECT

CSS TEXT EFFECT

CSS 2D TRANSFORM

CSS 3D TRANSFORM

CSS TRANSITION

CSS ANIMATION

**NAVIGATION BAR**

**DROPDOWN**

**CSS FORMS**

**CSS WEBSITE LAYOUT**

**CSS PAGINATION**

**MEDIA QUERY**

**USER INTERFACE**

**CSS RESPONSIVE**

**CSS GRID**