1.What Is CSS?

As we have mentioned, in this chapter we are going to be focusing on style, presentability, and the appearance of our web page. The main language used for styling a web page is CSS.

Fasten your seatbelts because in this chapter we are going to learn:

- What is CSS?
- How do you integrate CSS into HTML?
- How do you select an HTML element?
- How do you change the position of an element?

2. What Is CSS?

Cascading Style Sheets, commonly referred to as **CSS**, is a simple design language intended to simplify the process of making web pages presentable.

CSS determines the look and feel of a web page. With **CSS**, you can control the color of a text, the style of fonts, the spacing between paragraphs, and how columns are sized and laid out. It also defines what background images or colors are used, the layout designs, and, most importantly, the variations in display to adapt to different devices and screen sizes.

CSS is easy to learn and understand.It also offers powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup language HTML.

What is CSS And How It Works! - YouTube

https://www.youtube.com/watch?list=PL-w_yrNy8uTam5zYatOodamOW9-J9lHb_&v=-UxweAEaGqY&feature=emb_imp_woyt

CSS or Cascading Style Sheets is a Language used to style HTML pages. The Browser reads the CSS code and recognizes How the web page will be displayed. The CSS can manage the Color of an element, the backgrounds, its position … and even its events.

3.Linking CSS to HTML:

Before starting with CSS and how it works, we should first know how to link CSS to our HTML code.

There are three possible ways to attach CSS to HTML:

- Inline : by using the style attribute in HTML elements.
- Internal : by using a < style > element in the < head > section.
- External : by using an external CSS file.

Inline CSS:

The first way to style our HTML elements is via the attribute `style` where we can add the design we want to this element.

In situations where we have a small number of elements to deal with, this could be very helpful. But once we're dealing with large pages with lots of moving pieces, it will become extremely tedious to apply a separate `style` attribute to each element.

```
  <body>

      <h1 style="color: aqua">Hello World</h1>

      <p style="border: 1px blue; color: crimson">

          This is my first paragraph

      </p>

  </body>
```

Internal CSS:

The second way to add the CSS style is to include it directly inside the HTML document.
Notice the use of the `style` HTML element nested in the `head` element. The `style` element can be used to place CSS rules inline with an HTML document, like in the example below.
Although this is an absolutely valid way of adding CSS to your web pages, it will not let us reuse the same CSS style in other HTML files.

```
 <!DOCTYPE html>

<html lang="">

         <head>

                  <meta charset="" />

                  <meta name="viewport" content="width=, initial-scale=" />
```

```
                <!-- Internal style -->

                <style>

                        h1 {

                          color: aqua;

                          font: sans-serif;

                        }

                        p {

                          color: darkblue;

                          font: small;

                        }

                </style>

                <title></title>

        </head>

        <body></body>

</html>
```

External CSS

The last and probably the best way to use CSS (in terms of code reuse) is to create an external file with the extension **.css** and import it into our HTML document with the special link tag.

The `link` tag is placed inside the `head` of the document, with the attribute `href` that indicates the path of the CSS file.

```
<!DOCTYPE html>

<html lang="">

        <head>

                <meta charset="" />

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

                <meta name="viewport" content="width=, initial-scale=" />

                <title></title>

        </head>

        <body></body>

</html>
```

How to link css to html Quick Tutorial

https://www.youtube.com/watch?list=PL-w_yrNy8uTam5zYatOodamOW9-J9lHb_&v=Of8mr6QFkuo&feature=emb_imp_woyt

With external CSS, you can write the CSS code in an external file (with the .CSS extension) and import it in the <body>.

True

False

With internal CSS, you can add CSS code directly to your < body > section.

True

False

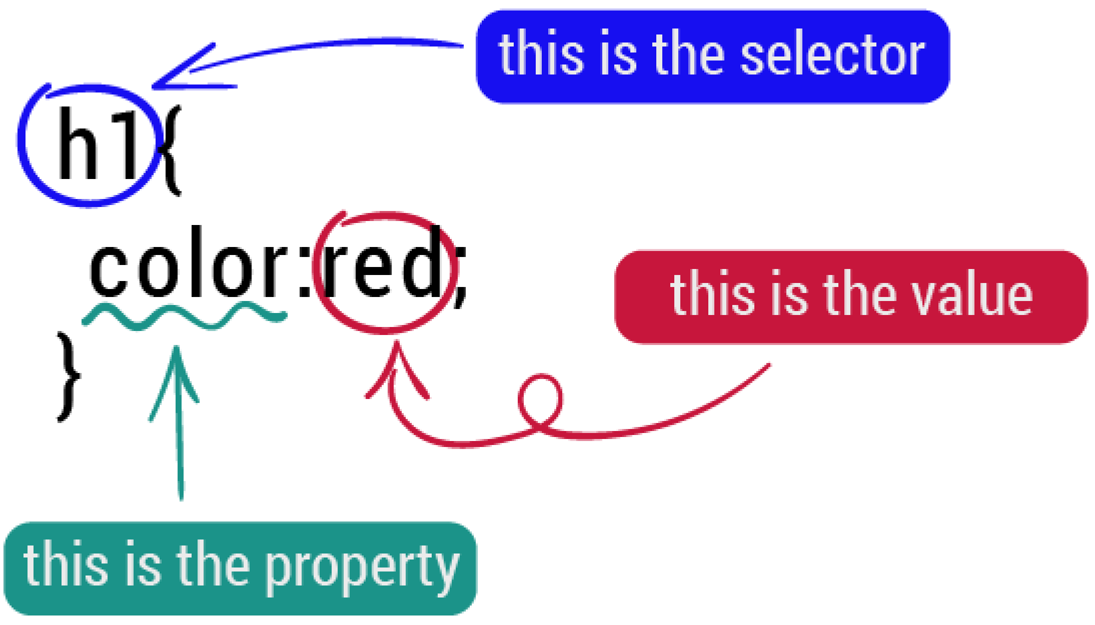
The external CSS file is imported using the tag link.

True

False

4.CSS Selector:

After attaching the CSS style, our next step is figuring out how to attach a chosen style to the proper element.
If we want to change the style of a particular element in our HTML page, we have to select that element first.
To do so, CSS provides a set of rules (each rule consists of a selector to indicate the element meant for modification) followed by a declaration block that contains a set of properties and those properties' values.



Selector Types:

CSS rules can select an element in a variety of ways. The three basic kinds of selectors are:

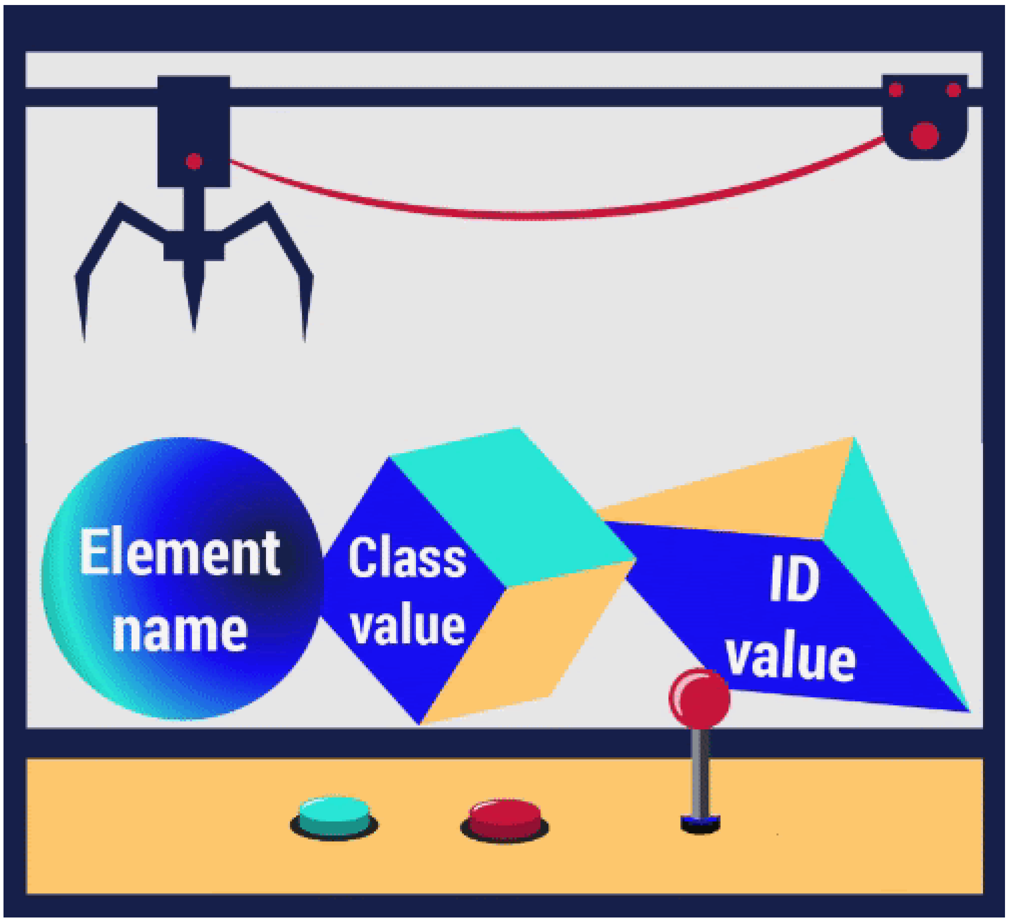
1. **Type** selectors: are used to select HTML elements by element name.
2. **Class** selectors: are used to select HTML elements by a specific class value.
3. **Id** selectors: are used to select an HTML element associated with a specific ID value.

There are other types of CSS selectors, but we are not going to cover them right now. You can find out more by following this link.

Combinators - Learn web development | MDN (mozilla.org)

https://developer.mozilla.org//en-US/docs/Learn/CSS/Building_blocks/Selectors/Combinators

Type Selectors are used to select HTML elements by element name. Class Selectors are used to select HTML elements by a specific class value. ID Selectors are used to select an HTML element associated with a specific ID value.



5.Type Selector:

In the previous slides, we have learned how to pick an HTML element in order to style it and we have encountered several types of CSS selectors.
Let's view the first type of selector and implement it in a real example.

Type:

Using a type selector is as simple as typing the name of the element. However, it’s preferable if you know when to use it. This selector is used when we want to apply the same rule on every element in the page.

For example, we want every `h1` title to be in red, the font will be `roboto` and the `text-align` will be in the center.

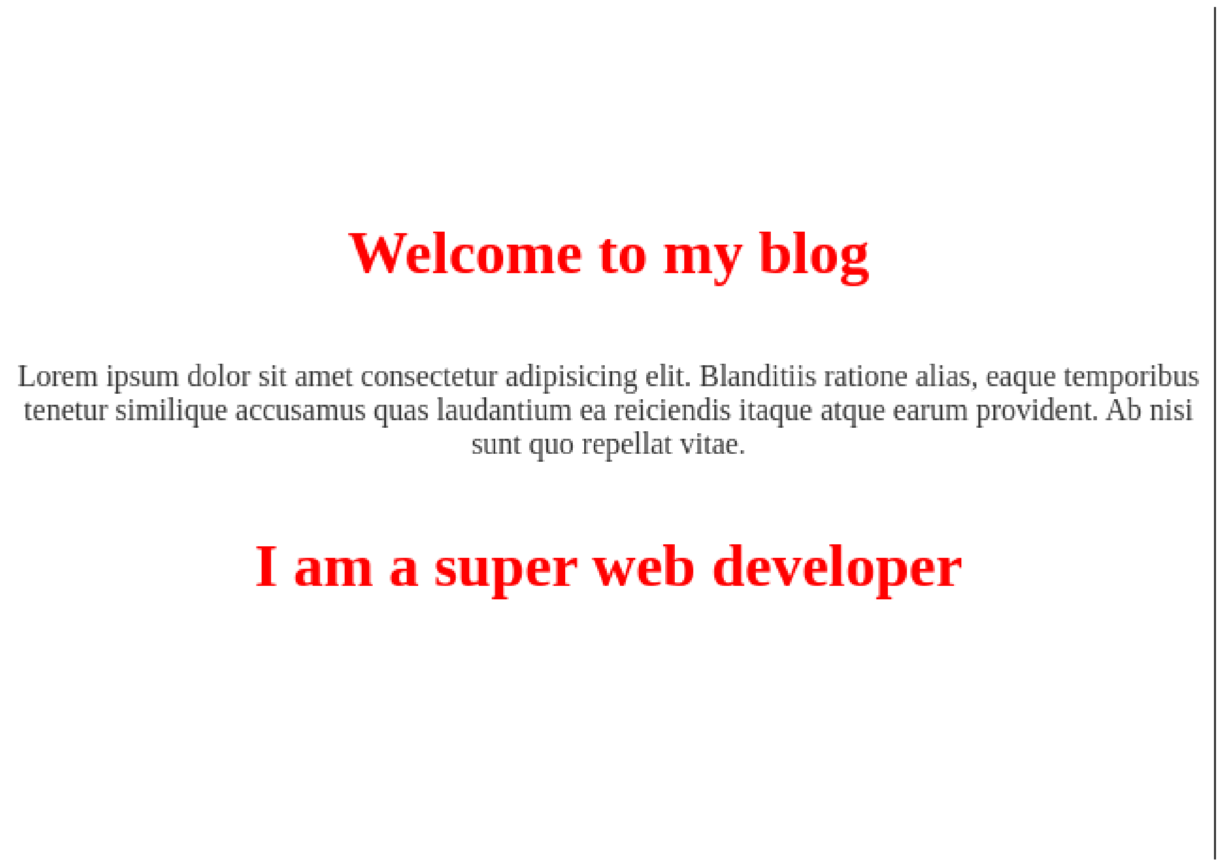
- HTML

```html
<!DOCTYPE html>

<html lang="">

        <head>

                <title>Type selector</title>

        </head>

        <body>

                <h1>Welcome to my blog</h1>

                <p>

                        Lorem ipsum dolor sit amet consectetur adipisicing elit.
Blanditiis

                        ratione alias, eaque temporibus tenetur similique accusamus
quas

                        laudantium ea reiciendis itaque atque earum provident. Ab
nisi sunt quo

                        repellat vitae.

                </p>

                <h1>I am a super web developer</h1>

        </body>

</html>
```

- CSS

```css
h1 {

        color: red;

        font-family: roboto;

        text-align: center;

}
```

- Output:



Welcome to my blog

Lorem ipsum dolor sit amet consectetur adipisicing elit. Blanditiis ratione alias, eaque temporibus tenetur similique accusamus quas laudantium ea reiciendis itaque atque earum provident. Ab nisi sunt quo repellat vitae.

I am a super web developer

Class Selectors:

The selection by type has worked fine until now, but let's say we have shared style not only for one type but for multiple types. For example, we are going to put all the text in a blue section and the other section will be in gray. In this case, the type selector is not extremely helpful. That is why we instead chose to use the **class** selector.

A class is a tag attribute that allows us to name an identifier for the element to adopt a certain style. You can specify a class on CSS with a (.) period

To use the `class` selector, you can follow the example below:

- HTML

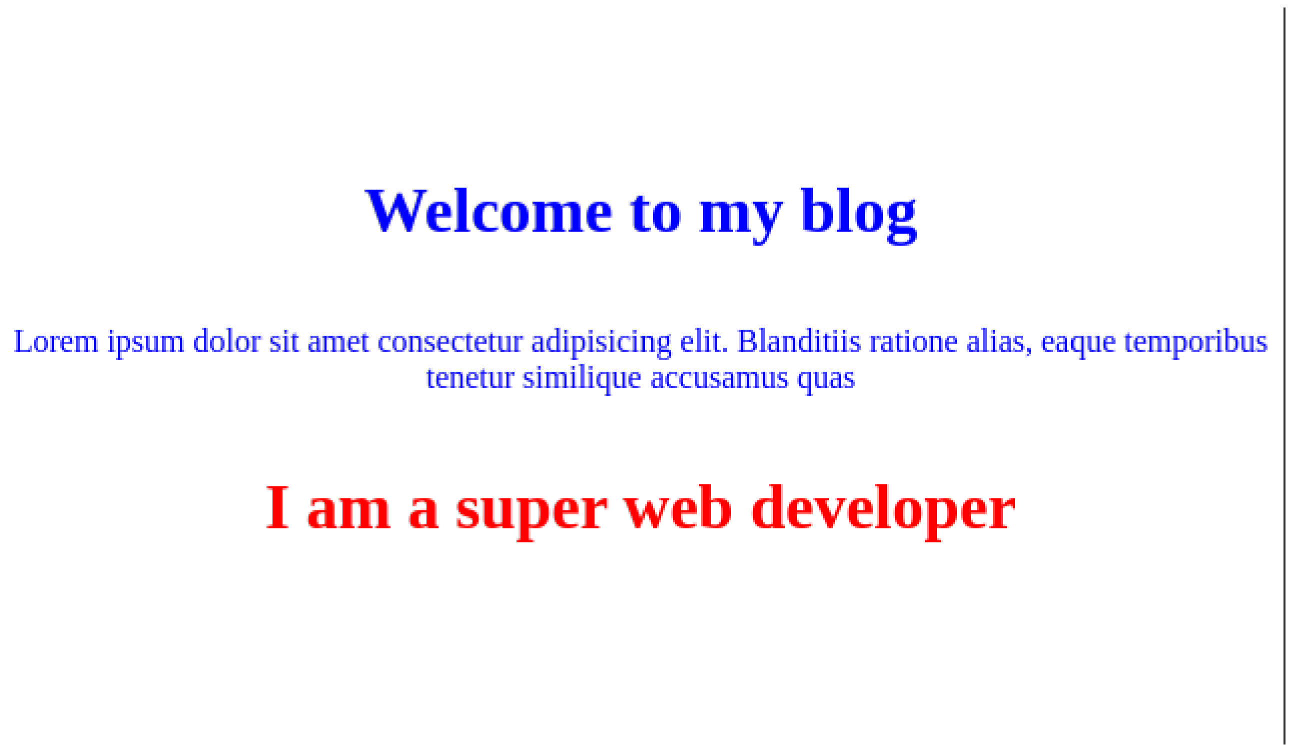
```
    <!DOCTYPE html>

<html lang="">

         <head>

                  <title>Type selector</title>

         </head>

         <body>

                  <h1 class="title">Welcome to my blog</h1>

                  <p class="title">

                           Lorem ipsum dolor sit amet consectetur adipisicing elit.
Blanditiis
```

```
                           ratione alias, eaque temporibus tenetur similique accusamus
quas

                  </p>

                  <h1>I am a super web developer</h1>

         </body>

</html>
```

- CSS

```
.title{

  color:blue

}
```

- Output:


Welcome to my blog

Lorem ipsum dolor sit amet consectetur adipisicing elit. Blanditiis ratione alias, eaque temporibus tenetur similique accusamus quas

I am a super web developer

ID Selector:

The final type of CSS selector is the selection by ID.

ID selectors in CSS allow you to target elements (Tags) by their ID values. ID selectors are unique, so you can only apply them to the content of one element. To reference an ID, you precede the ID name with a hash mark (#).

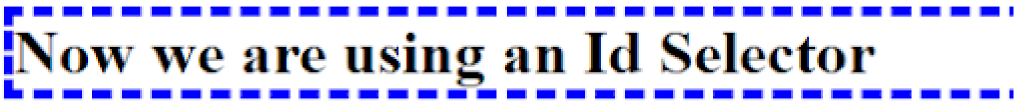
- HTML

```
     <h1 id="blue-bordered">Now we are using an Id Selector</h1>
```

- CSS

```
#blue-bordered {
    border: 5px dashed blue;

}
```

- Output:



Multiple Selectors :

One of the most important principles in programming is the “DRY” principle: Don’t repeat yourself.

Let's assume that we have a `p` tag, a `div` tag, and a `section` tag that share the same background color. So, by following the “DRY” principle, we must not repeat the styling three times.

CSS allows us to do that. We can select multiple elements by separating the selectors with commas, like this:

```
/* Selecting multiple HTML element types */

       h1, p{

           border: 1px solid black;

       }


      /* Selecting styles to be applied to several classes */

      .FirstItem, .LastItem {

          font-size: 1.2em;

      }


      /* Using multiple Kinds of selectors */

      h3, .red, #redElement {

          color: red

      }
```

h1 is a Type Selector. .Borderedis a class selector. #blue-bordered is an ID Selector
. To apply the same style on multiple Selectors we separate them with a Comma.

6.CSS selectors : combinators

Descendant Combinator:

The descendant combinator is typically represented by a single space character. It combines two selectors so that elements matched by the second selector are selected if they have an ancestor (parent, parent's parent, parent's parent's parent, etc.) element matching the first selector. Selectors that utilize a descendant combinator are called descendant selectors.

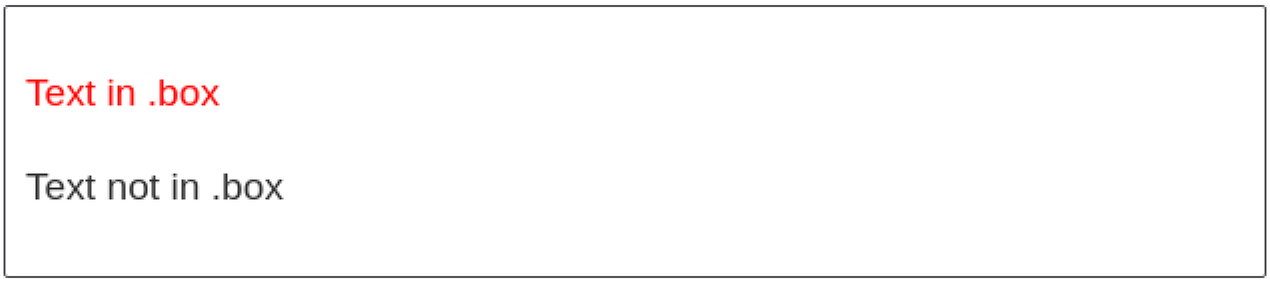
- HTML

```html
<div class="box"><p>Text in .box</p></div>

<p>Text not in .box</p>
```

- CSS

```css
         .box p {

                          color: red;

         }
```

- Output


Text in .box

Text not in .box

Child Combinator:

The child combinator (>) is placed between two CSS selectors. It only matches those elements matched by the second selector that are the direct children of elements matched by the first. Descendent elements further down the hierarchy don't match.
For example, to select only `<p>` elements that are direct children of `<article>` elements:

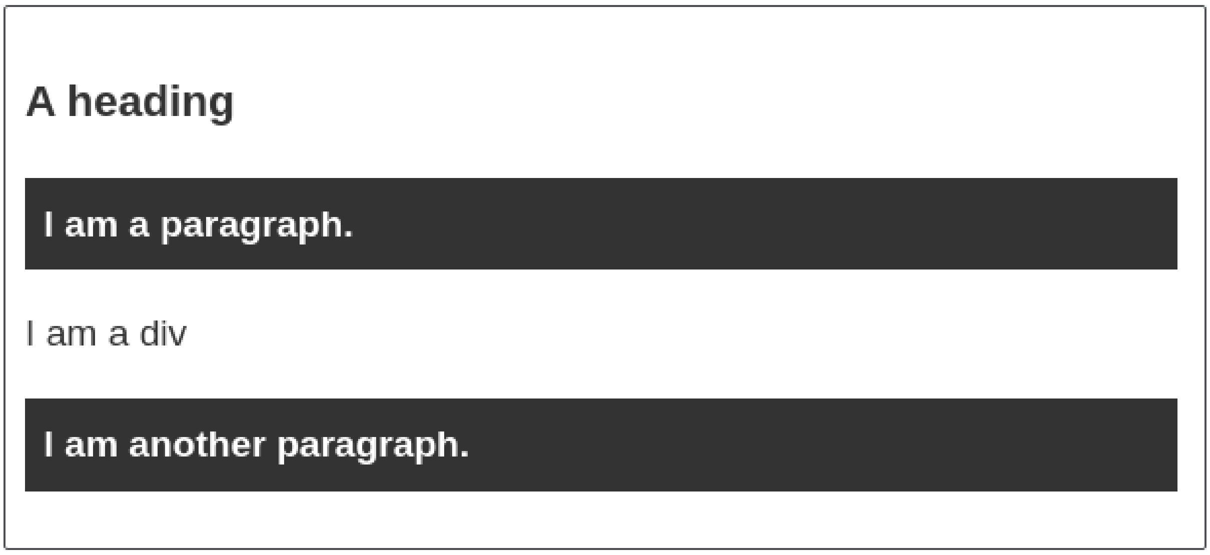
- HTML

```html
<article>

    <h1>A heading</h1>

    <p>I am a paragraph.</p>

    <div>I am a div</div>

    <p>I am another paragraph.</p>

</article>
```

Css

```css
Article > p {

    font-weight: bold;

    background-color: #333;

    color: #fff;

    padding: .5em;

}
```

- Output


A heading

I am a paragraph.

I am a div

I am another paragraph.

Adjacent Sibling Combinator:

The adjacent sibling selector (+) is used to select something if it is right next to another element at the same level of the hierarchy.
For example, to select all `<p>` elements that come right after `<h1>` elements:

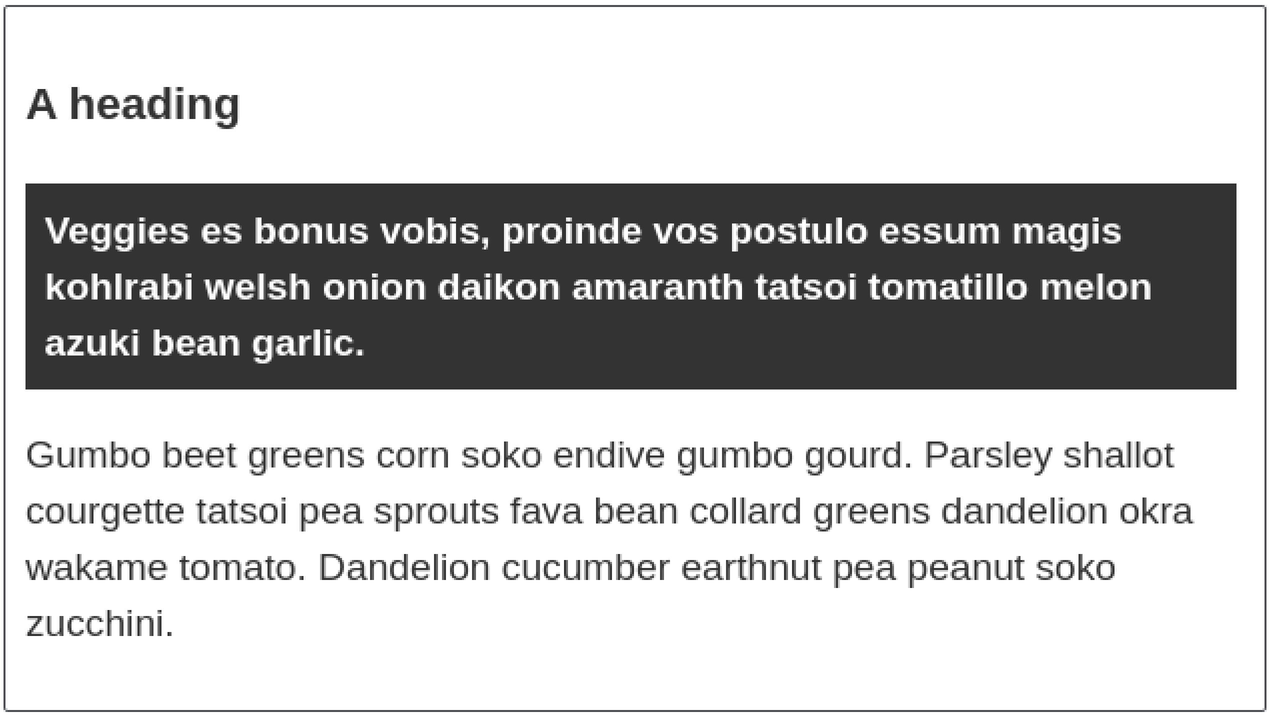
- **HTML** :

```
<article>

    <h1>A heading</h1>

    <p>Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion
daikon amaranth tatsoi tomatillo

            melon azuki bean garlic.</p>



    <p>Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi
pea sprouts fava bean collard

            greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut
soko zucchini.</p>

</article>
```

- **CSS**:

```
h1 + p {

    font-weight: bold;

    background-color: #333;

    color: #fff;
    padding: .5em;
}
```

- Output:


A heading

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

General Sibling Combinator

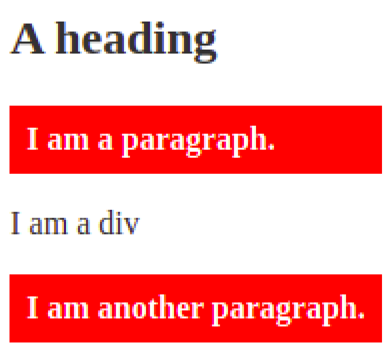
If you want to select siblings of an element even if they are not directly adjacent, then you can use the general sibling combinator (~). To select all `<p>` elements that come anywhere after `<h1>` elements, we'd do this:
HTML

```
<article>
    <h1>A heading</h1>
    <p>I am a paragraph.</p>
    <div>I am a div</div>
    <p>I am another paragraph.</p>
</article>
```

CSS

```
h1 ~ p {
    font-weight: bold;
    background-color:red;
    color: #fff;
    padding: .5em;
}
```

Output


A heading

I am a paragraph.

I am a div

I am another paragraph.

To select only the children of a certain parent element you ...

Must indicate the parent element and then the child element, with a > bracket between them.

Must indicate the child element and then the parent element, with a > bracket between them.

The descendant combinator is represented by a

Single space

Dot

Comma

The ~ operator represents the:

Descendant Combinator

Direct child combinator

Adjacent sibling combinator

General sibling combinatory

Css selectors : Pseudo classes

CSS Pseudo classes:

Have you ever tried to put your cursor on a button and its color changed? Or have you ever clicked on a link and the same thing happened? We call this “marked as clicked” and it can be done with CSS pseudo-classes.

As you can see, the structure is clear. You name your SELECTOR, add `:` then define your action.
Next, we’ll learn a few awesome actions you can do using CSS pseudo-classes. Try them on your editor.

```
/* unvisited link */

a:link {

    color: tomato;

}


/* visited link */

a:visited {

    color: tomato;

}


/* mouse over element */

div:hover {

    color: tomato;

}


/* selected link */

a:active{

    color: #0000FF

}
```

Pseudo-classes let you apply a style to an element not only in relation to the content of the document tree, but also in relation to external factors like the history of the navigator (:visited for example), the status of its content (like:checkedon certain form elements), or the position of the mouse (like :hover, which lets you know if the cursor is over an element or not).

CSS Box Model

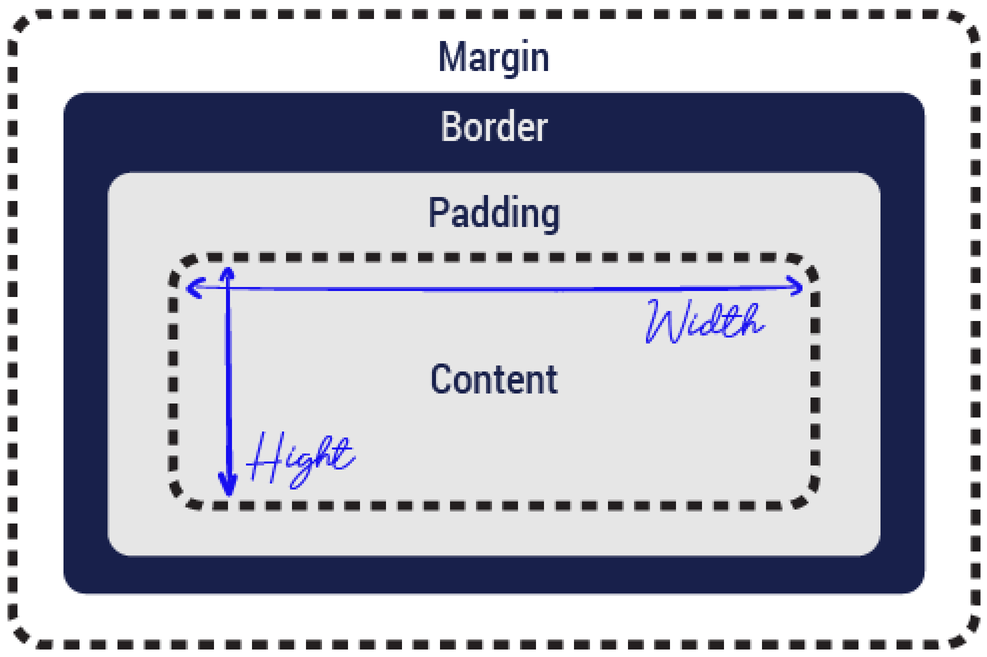
Every element in web design is a rectangular box (even if it is a circle).

The CSS box model is defined by the four layers below:

- Content: Is where text and images appear.
- Padding: Is transparent. It clears area around the content.
- Border: Goes around the padding and content. We have already seen it earlier.
- Margin: Clears an area outside the border. The margin is transparent.

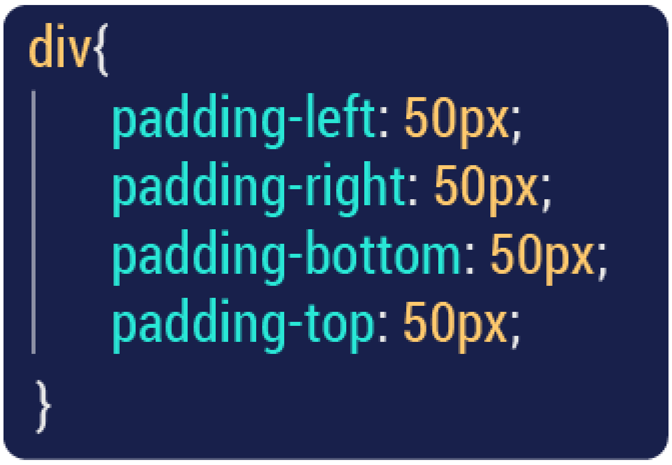
Also, we can determine the dimension of any html element through the `width` and the `height`

We'll get to know them more individually later.

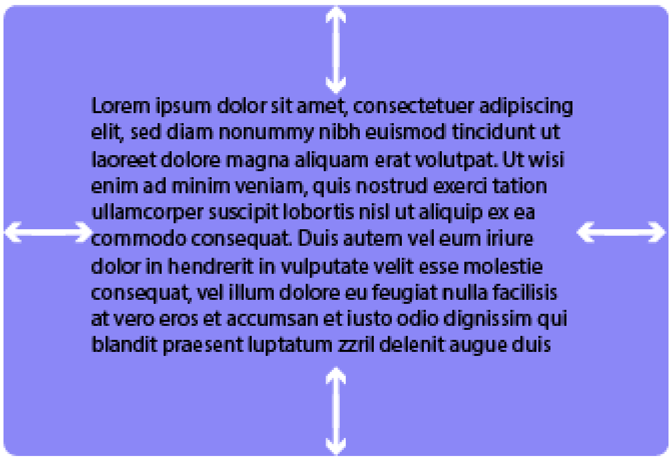


Padding

- We use the padding property to create spacing between an element's content area and border. The padding property applies this spacing in different ways depending on how many values you provide.
  The Padding is the INSIDE of the element.
  Suppose we have a div with many HTML elements inside. The problem is, we want to move the elements inside without moving the div itself. How can we accomplish that? Padding is our hero here!



```
div{
    padding-left: 50px;
    padding-right: 50px;
    padding-bottom: 50px;
    padding-top: 50px;
}
```



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis

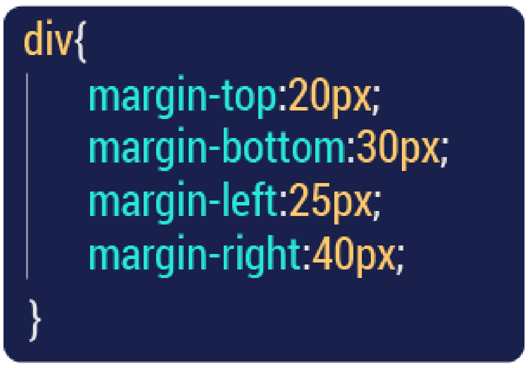
- We can treat the padding value separately as we have seen in the previous example, or we can make it in one line like the following code.

```
.box {

  padding: <padding-top> || <padding-right> || <padding-bottom> || <padding-left>

}
```

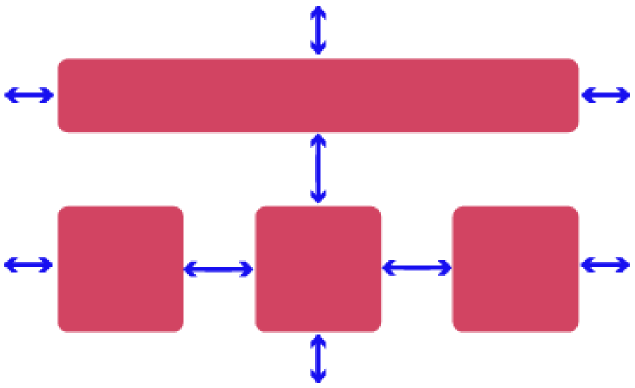
Margin

As a first step, go back to the CSS box model scheme to see where the margin is positioned.

Basically, margins allow us to move our tag wherever we want, in any direction: top, bottom, right, or left.
The Margin is the OUTSIDE of the element.
The margin property is very similar to the padding property, except it allows you to define the spacing around the outside of an HTML element past the border. Like padding, it allows you to define single or multiple values.



```
div{
    margin-top:20px;
    margin-bottom:30px;
    margin-left:25px;
    margin-right:40px;
}
```



Height & Width

As we have seen, every element in HTML is a rectangular box. By the laws of geometry, every rectangle is characterized by its width and height.

We can alter these two properties of any HTML content element by using the CSS attribute width or height.

One last thing to always keep in mind:
`Total_width= width+padding_right+padding_left+margin_right+margin_left`
To change the value of these properties, just follow the following example:

- HTML

```
      <div>

         <p>I want to get bigger</p>

      </div>
```

- CSS

```
div{

    height: 150px;

    width: 90%;

}
```

Border

Like its name implies, the border CSS property sets the border of an element.The border property is a shorthand syntax in CSS that accepts multiple values for drawing a line around the element it is applied to.

```
border:  <border-width> || <border-style> || <color>
```

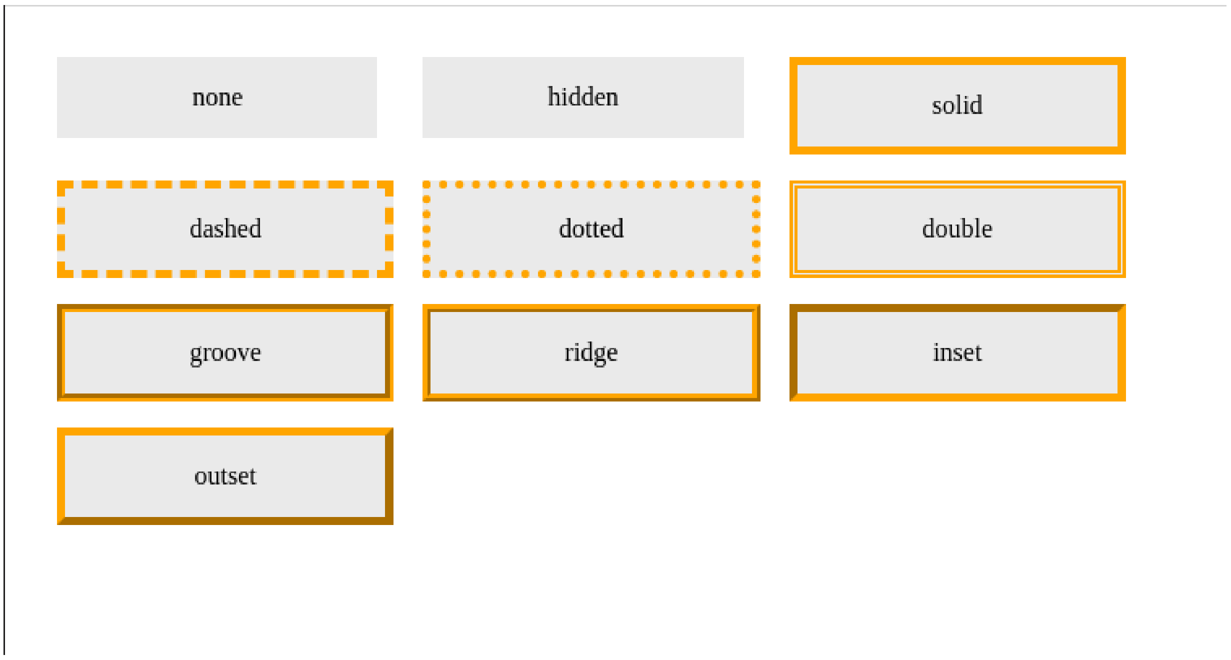
- `border-width`: specifies the thickness of the border.
    - <length>: A numeric value measured in px, em, rem, vh and vw units.
    - thin: the equivalent of 1px
    - medium: the equivalent of 3px
    - thick: the equivalent of 5px
- border-style: Specifies the type of line drawn around the element, including:
    - solid: a solid, continuous line.
    - none (default): No line is drawn.
    - hidden: a line is drawn, but not visible. This can be useful for adding a little extra width to an element without displaying a border.
    - dashed: a line that consists of dashes.
    - dotted: a line that consists of dots.
    - double: adds two lines around the element
    - groove: adds a bevel based on the color value in a way that makes the element appear pressed into the document.
    - idge: similar to groove, but reverses the color values in a way that makes the element appear raised.
    - inset: adds a split tone to the line that makes the element appear slightly depressed.
        - outset: similar to inset, but reverses the colors in a way that makes the element appear slightly raised.
- color: specifies the color of the border and accepts `<rgb()>`, `<rgba()>`, `<code><hsl()>`, `<hsla()>`, `<hex-color>`, `<named-color>`

- CSS

```
.box-1 {

  border: none;

}



.box-2 {

  border: 5px hidden red;

}



.box-3 {

  border: 5px solid orange;

}



.box-4 {

  border: 5px dashed orange;

}



.box-5 {

  border: 5px dotted orange;

}
```

```
.box-6 {
  border: 5px double orange;
}

.box-7 {
  border: 5px groove orange;
}

.box-8 {
  border: 5px ridge orange;
}

.box-9 {
  border: 5px inset orange;
}

.box-10 {
  border: 5px outset orange;
}
```

- Output:



Border Radius

We can give any element "rounded corners" by applying a border-radius through CSS. You'll only notice if there is a color change involved. For instance, if the element has a background-color or border that is different from the element, it's incorrect. Simple examples:

- HTML

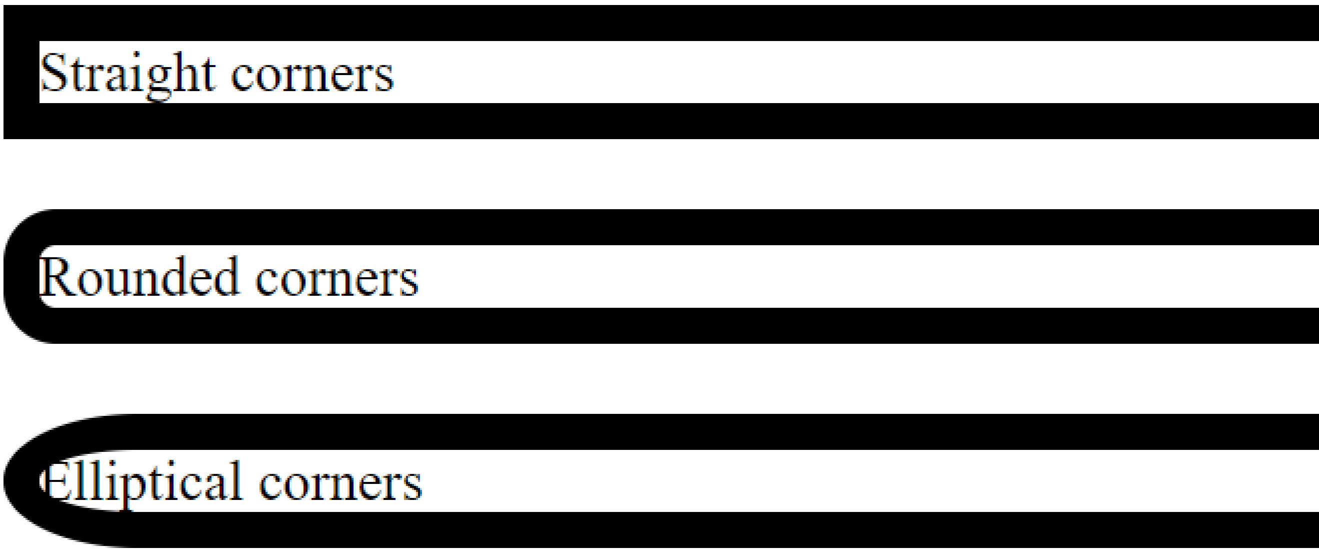
```
<div>

  <p>Straight corners</p>

  <p class="rounded">Rounded corners</p>

  <p class="elliptical">Elliptical corners</p>

</div>
```

- CSS

```
  p {

    border: 10px solid black;

    margin: 20px;

  }



  .rounded {

    border-radius: 15px;

  }
```

```
.elliptical {

  border-radius: 50px / 25px;

}
```

- Output:



The padding property applies the spacing in different ways depending on how many values you provide.

True

False

The value of a padding is expressed in:

%

Px

Vw / vh

Em

All above

Margin is the OUTSIDE of the element?

True

False

Margin can be applied only on the div tag

True

False

Width and height are the dimensions of the CSS tag, you can only use px to represent them.

True

False

Using the % values in width makes it relative to the containing block width

True

False

If you want to create borders that have rounded corners, use the

border-radius property

border-rading property

border property

If you want to create dotted borders use the

border: double

border:dotted

border:dashed

The border radius can accept only one value

True

False

Font Styling:

To style the font, CSS gives us a large set of properties. We'll be studying a few of them:

Font-family

It helps us change the font family of an HTML element, we set one or more fonts for this property .

```
font-family: Helvetica neue, roboto;
```

Color

It changes the text color, it accepts a named color, hexadecimal value, and rgb value.

```
 color: rgb(0, 0, 255);

  /* same as

  color: blue;

  color: #0000ff;

  color: #00f;

  color: hsl(0, 100%, 50%); */
```

Font-style

It's used to turn italic text on and off. Possible values are as follows (you'll rarely use this, unless you want to turn some italic styling off for some reason).

```
 font-style: bold;

/*

other possible value

 font-style:normal;

 font-style:italic;

 font-style:oblique;

*/
```

Font-weight
It sets how bold the text is. This has many values available in case you have many font variants available (such as -light, -normal, -bold, -extrabold, -black, etc.). Realistically, you'll rarely use any of them except for normal and bold.

```
 font-weight: normal;

/*

other possible value

 font-weight:bold;

 font-weight:bolder;

 font-weight:lighter;

 font-weight:100-900

*/
```

Text-decoration

It sets or unsets text decorations on fonts (you'll mainly use this to unset the default underline on links when styling them). Available values are:

```
 text-decoration: none;
```

```
/*

other possible value

 text-decoration:underline;

 text-decoration:overline;

 text-decoration:line-through;

*/
```

Text Layout Styles:

With basic font properties out of the way, let's now have a look at properties we can use to affect text layout.

The text-align
It's used to control how a text is aligned within its containing content box.
The available values are as follows, and work in pretty much the same way as they do in a regular word processor application:
Left: Left- aligns the text
Right: Right- aligns the text
Center: Centers the text
Justify: Makes the text spread out

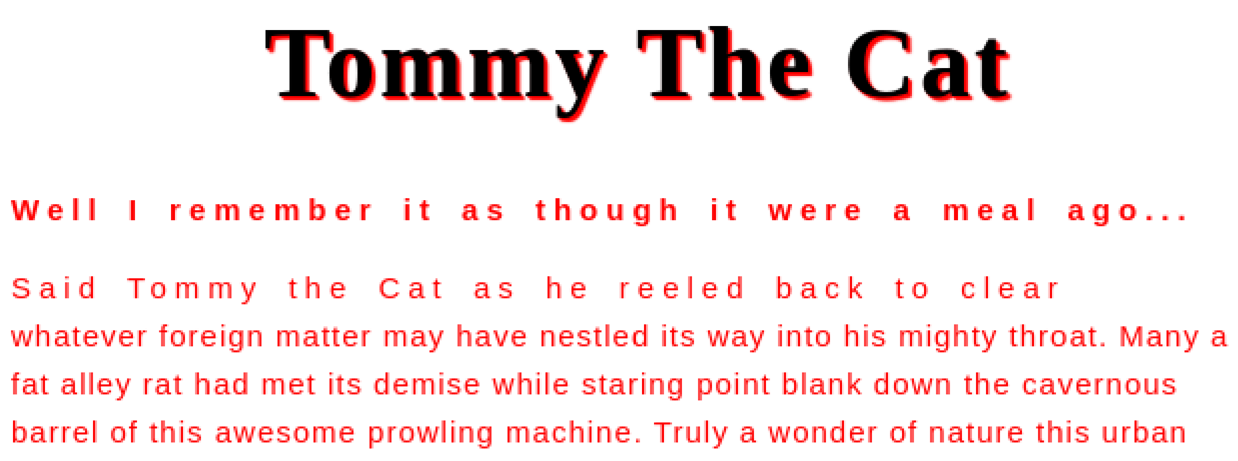
Line-height
It sets the height of each line of text — this includes most length and size units, but can also take a unitless value, which acts as a multiplier and is generally considered the best option. The font-size is multiplied to get the line-height.

Letter-spacing and word-spacing
They allow you to set the spacing between letters and words in your text. You won't use these very often, but might find a use for them to get a certain look, or to improve the legibility of a particularly dense font. They can use most length and size units.

CSS Output :

```
first-line {

  letter-spacing: 4px;

  word-spacing: 4px;

}
```

The code output would be :



Tommy The Cat

Well I remember it as though it were a meal ago...

Said Tommy the Cat as he reeled back to clear whatever foreign matter may have nestled its way into his mighty throat. Many a fat alley rat had met its demise while staring point blank down the cavernous barrel of this awesome prowling machine. Truly a wonder of nature this urban

Text Size:

There will be many instances where you will want to change the default size of text elements. The size of your text can be changed using the font-size property. The font-size takes both absolute and relative values. The most common absolute value is px and the most common relative values are ems and rems.
em and rem units are both relative measurement values that work similarly to percentages. They serve as a multiplier in reference to some other unit of measurement.
For font-size:

- 1 em is equivalent to the font-size of the element's parent.
- 1 rem is equivalent to the font-size of the root element of the entire HTML document.

```
h1{

    font-size: 50px

}
```

You can change the font of your text using :

the font-family property

the family-font property

the family-style property

If you want the text to spread out to fill out the full width of its container use the:

text-align:justify

text-align:center

The color value can only be a hexadecimal value.

True

False

CSS Display:

Every element on a web page is a rectangular box. The display property in CSS determines just how that rectangular box behaves.

The CSS display is most useful when :

- we need to align two HTML elements together in the same line.
- we want to display an HTML element as a block.
- we want to hide an HTML element.

In this lesson, we will see 4 types of CSS display: none, inline, block and inline-block.

Note: The default display of an element is inline or block.

The default display of an element is:

inline or block.

none.

Which of these is not a display type in CSS?

inline

block

hover

The display property controls when the element is shown.

True

False

Display: none

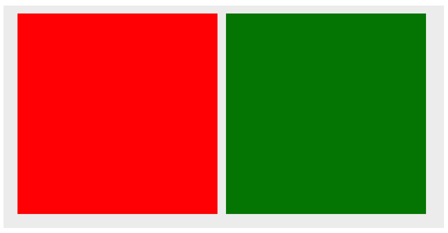
We can hide elements by declaring a display: none value. Another way is to declare visibility: hidden instead of display: none, but there is a difference between them..

To show the difference, let’s hide one of the boxes below:

- CSS

```
#box-2 {

  display: none;

  width: 100px;

  height: 100px;

  background: blue;

}
```

- Output

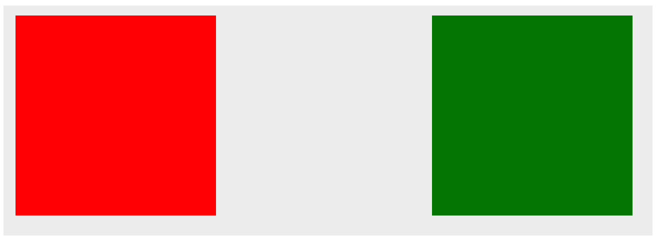


Display: none removes an element from the view

- CSS

```
#box-2 {
  width: 100px;

  height: 100px;

  background: blue;

  visibility: hidden;

}
```

- Output



Blue box is invisible now, but it's still there

None Turns off the display of an element so that it has No Effect on layout (the document is rendered as though the element did not exist). All descendant elements also have their display Turned Off

. To have an element Take Upthe space that it would normally take, but without actually rendering anything, use the Visibility property instead.

Display: inline

Like the name indicates, this feature enables you to put specified HTML elements in the same line. Let's understand it with an example.

- HTML

```
      <p>One</p>

      <p>Two</p>

      <p>Three</p>
```

- CSS

```
      p {

        display: inline;

      }
```

- Output

One Two Three

In an Inline formatting context, Boxes are laid out Horizontally, one after the other, beginning at the Top of a containing block. Horizontal margins, borders, and padding are respected between these boxes.

Display: block

- HTML elements are divided into default block elements (automatically force a line-break without the use of `<br>`) and inline elements (need `<br>` to line-break).

- Block-level elements:

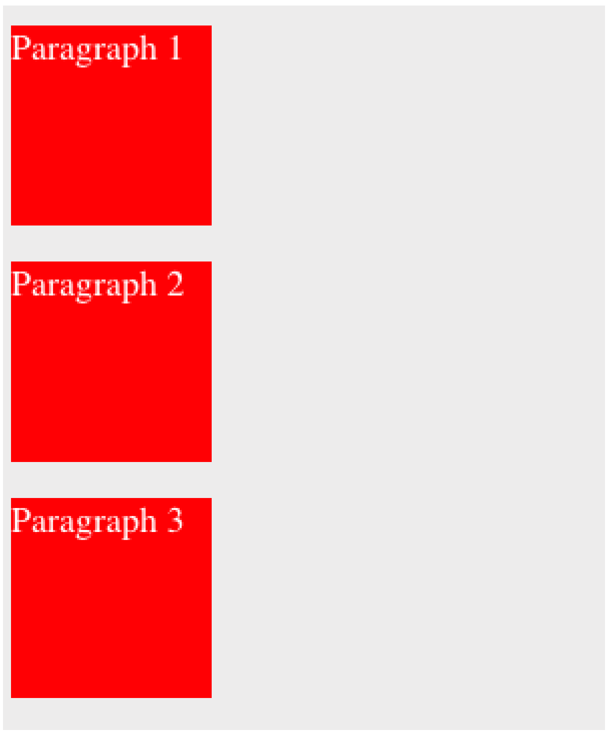
    - take full-width (100% width) by default.
    - each get displayed in a new line.
    - allow width & height properties to be set.
    - can contain other block or inline elements.

What about forcing an inline element to act like a block?

- CSS

```
div{

display: block;

}

     p {

  height: 100px;

  width: 100px;

  background: red;

  color: white;

}
```

- Output



In a Block formatting context, boxes are laid out one after the other, Vertically , beginning at the top of a containing block. The vertical distance between two Sibling Boxes is determined by the 'Margin' properties. Vertical margins between adjacent Block-Level boxes collapse in a block formatting context.

Display: inline-block

In some cases, both of the display values may not be enough for better web design. In that case, a third display behavior, display: inline-block, comes to the rescue and makes alignment much easier.
As we can understand from its name, display: inline-block declaration shows both the characteristics of inline and block-level elements.
In other words, we can think of an inline element with width & height properties that can be set, or we can think of a block-level element that doesn't have to start with a new line.

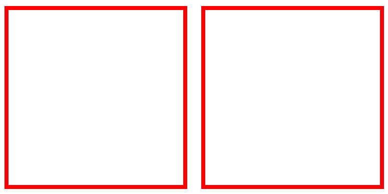
- HTML

```
    <div class="test"></div>

    <div class="test"></div>
```

- CSS

```
      .test {

        width: 50px;

        height: 50px;

        border: 1px solid red;

        display: inline-block;

      }
```

- Output



Display:none : hides the element with the correspondent selector Display:inline:enables you to put the specified HTML elements in the same line. Display:inline-Block :displays blocks on the same line.

Conclusion

What you should know by now :

https://www.youtube.com/watch?time_continue=10&v=z8R8AZI4JuY&feature=emb_logo

css Project

Objective

Remember that portfolio we have created in the last checkpoint? Now is the time to give it some style.

Instructions

1. *Create a file `styles.css`.*
2. *Link it to our HTML project.*
3. *Change the display of the navbar to make it inline.*
4. *Change the font to roboto.*
5. *Add classes attributes to HTML / HTML documents .*
6. *Be creative !*

- ***Hint:*** *You can use the Internet to search for portfolio models.*