






# HTML + CSS

|   |  |
|---|--|
|  Author        |  Jay Vachhani |
|  Category      | HTML/CSS   |
|  Date Started  | @Jun 25, 2020  |
|  Last Revision | @Jul 6, 2020   |

## Acknowledgement:

I would like to personally thank all of the open source community in their continued effort to invent and innovate for the greater good of technology.

I have used multiple sources in writing this document. From online courses/articles/websites to also my own personal learning and knowledge. By no means am I claiming to be the sole creator of all of the information displayed here, It is a collection of curated material from myself and the wider web. I have written this document out myself, should you feel there are any major inconsistencies or inaccurate information being presented here, please contact me using the tab "Contact Me" on my website [jayvachhani.tech](http://jayvachhani.tech) , please use the name of this document in your message.

---

## Basic HTML and HTML5

### HyperText Markup Language → HTML

HTML is used to describe the structure of a web page. It can be imagined as the bricks upon which a house is built. It uses a unique syntax and notation to structure information about a webpage to the browser.

Elements, such as headings, use `<h1>This is a heading</h1>` opening and closing tags.

HyperText existed in the first webpages and was the main use case of these webpages. Pages contained static documents which contained references to other documents. These references contained hypertext links which the browser would use to navigate to the referred document.

The W3 consortium updates the HTML specification to ensure that any webpage/app can be supported on any browser. HTML5 is the latest spec.

## Structure

All HTML Documents have to have a Doctype declared as its first line of code. This tells the browser which version of HTML you are using to code this webpage. using `<!DOCTYPE html>` tells the browser you are using HTML5, this is followed by `<html>` tags which will contain the code.

```
<!DOCTYPE html>
<html>
<html lang="en">
  <head>
    <meta charset="utf-8">

    <title> This is a Title </title>
    <link rel="stylesheet" href="css/styles.css?v=1.0">
  </head>

  <body>
    <script src="js/scripts.js"></script>
    <!-- Our 1 Million lines of HTML Code will go here -->
  </body>

</html>
```

## Elements

All tags surrounding elements are lowercase.

`<h1>lowercase</h1>` → YES

`<H1>UPPERCASE</H1>` → NO

Uppercase tags will not function in browsers.

## Comments

Comments are used to inform developers working on the code about the code and its functions. Comments in HTML require `<!-- (insert comment here) -->` tags. It proves useful to comment out code for testing purposes.

## Headings

There are 6 levels of headings in HTML, h1 → h6. They each denote a different level with `<h1>` being the largest and `<h2>` `<h3>` ... `<h6>` each getting progressively smaller

```
<h1> Title of the Page </h2>
<h2> Sub Heading level 1 </h2>
```

## Paragraphs

p elements are utilised for paragraph texts on websites and are denoted by `<p>` `</p>` tags

```
<p> This is a very long sentence which could be considered a paragraph by a writer</p>
```

Lorem Ipsum by Cicero of Ancient Rome is used as standard placeholder text by web developers.

## HTML5 Elements

HTML5 includes new elements which assist SEO (Search Engine Optimisation) and accessibility. They also help navigation and structure of a page. These elements include:

`header` `footer` `nav` `video` `article` `section` `main`

```
<!-- These elements act as containers for child elements within -->
<main>
  <h1> This is the main container </h1>
  <p> These are child elements within the main tags </p>
</main>
```

## Images

Images are a very useful tool for websites and can be found on nearly all that exist currently. They are declared using their URL within a `src` (source) **attribute** and a `alt` attribute which describes the image.

```

```

Note that **image tags** are **auto closing** and do not require `</img>` tags

## Anchors

An anchor element is used to add URL links for content that isn't on the webpage. They use a `a` tag and a `href` (hypertext reference) attribute which preposes the destination URL. The `href` can be used to link the `id` of other internal sections too.

```
<a href="this.sure.is.a.URL.to.a.website">this definitely links to a website</a>
<a href="#section-header">Jump to Section</a> <!-- Note the use of "#"-->
...
<h3 id="section-header">Section</h3>
```

It can display specific text you want for the link to attached to.

You can set `href="#"` to add a link later.

## Nested Anchors

anchors can be nested within a paragraph for embedded links in the content of the page.

```
<p>Follow <a target="_blank"
href="https://Link.io">this</a>
link.</p>
```

This would look like : "Follow this link.". The `target` attribute is set to `"_blank"` which means to open a new tab when this link has been clicked.

## Image Anchors

Sometimess you want someone to click on a picture to travel to a new page. Adding the image element into the `a` tags can achieve this.

```
<a href="#"></a>
```

## Lists

Lists are a useful tool to deliver information. HTML can employ a variety of list structures which serve different purposes.

### Unordered Bulleted list

These are declared using `ul` tags and each child element within is opened and closed using `li`

```
<p>This looks like:</p>
<ul>
  <li>Could be a bird</li>
  <li>Could be a plane</li>
  <li>Probably isn't a dude</li>
</ul>
```

This looks like:

- Could be a bird
- Could be a plane
- Probably isn't a dude

## Ordered Numbered List

These are declared using `ol` tags and each child element within is opened and closed using `li`

```
<p>This looks like:</p>
<ol>
  <li>Orchestra</li>
  <li>Flamingo</li>
  <li>Drum & Bass</li>
</ol>
```

This looks like:

1. Orchestra
2. Flamingo
3. Drum & Bass

## Web Forms

Web forms are used, nowadays, by virtually all websites for many purposes. They allow input data from the user to be sourced and worked with. Social media sites use forms to add content, whilst others use forms to create a personalized experiences. An easy example is a 'Contact Us' form which organisations use on their sites.

### Forms

These allow users to submit data which will change the the webpage for the user specifically. They are declared with `form` tags with an `action` attribute which describes the user action.

A text box for user input is created using the `input` tag, with a `type` attribute, which is self closing. The tag also allows for a `placeholder` which disappears when user begins to add input. Adding a `required` tag to the type will make sure user can't submit without this being satisfied.

```
<form action="/submit-link">
  <input type="text"required
  placeholder="Add Link Here">
  <button type="submit">Submit</button>
</form>
```

This looks like:



Finally we declare a button using `button` tags with a `type` attribute relating to button function. Now our form is complete

## Radio Buttons

These are used for multiple choice entries where you want only one answer out of multiple options

They are declared as a nested element within `label` tags with a `for` attribute. Then comes an `input` tag with `id`, `type` and `name` attributes, more can be added depending on situation.

A `value` tag is added to the input if data needs to be sent to a server, it is sent through the value attribute, otherwise it would send a default value of on if the user has selected any of the options.

```
<form>
  <label for="Bird">
    <input id="Bird" value="Bird" type="radio"
      name="bird-plane-dude">Bird
  </label><br>
  <label for="Plane">
    <input id="Plane" value="Plane" type="radio"
      name="bird-plane-dude">Plane
  </label><br>
  <label for="Dude">
    <input id="Dude" value="Dude" type="radio"
      name="bird-plane-dude">Dude
  </label><br>
</form>
```

☐ Bird  
☐ Plane  
☐ Dude

☐ Bird  
☒ Plane  
☐ Dude

Best practice denotes that we assign the `for` and `id` tags with the same name so assistive technologies can create a relationship with the label and child input element. Adding a `<br>` assists in vertically structuring the check boxes.

## Check Boxes

These are declared similarly to Radio Buttons except their `type` attribute is set to `checkbox`. If we wanted to check these boxes by default we use the `checked` attribute at the end of the `input` tag.

```
<form>
  <label for="Bird">
    <input id="Bird" value="Bird" type="checkbox"
      name="bird-plane-dude">Bird
  </label><br>
  <label for="Plane">
    <input id="Plane" value="Plane" type="checkbox"
      name="bird-plane-dude" checked>Plane
  </label><br>
  <label for="Dude">
```

☐ Bird  
☐ Plane  
☐ Dude

```
<input id="Dude" value="Dude" type="checkbox"
name="bird-plane-dude" checked>Dude
</label><br>
</form>
```

☐ Bird  
☒ Plane  
☒ Dude

These are useful because they allow for multiple options to be selected, unlike the radio buttons.

## Division

We saw that HTML5 has a variety of tags for page structure, the `<div>` tags act as a universal divider for content you want to be grouped together within parent page structure tags like `main` or `header` for example.

```
<main>
  <div>
    <p> This paragraph is contained in a set of division tags.</p>
  </div>
</main>
```

# CSS

## Basic CSS

Cascading Style Sheets (CSS) are utilised to communicate how to display the content of your html document. They are one of the most common ways to make your HTML page standout, and the more time given to your CSS will often lead to a better user experience of your website. Like HTML, it has also been adapted by all major browsers.

We will look at all the aspects it give you control over, but here are some quick names:

Color + Font + Position + Space + Size+ Decoration +Transition

### Approaches to CSS Styling:

1. Inline Styling → useful for learning

This is when HTML elements are directly styled with attributes in their opening tags

2. Class & Selector Styling → useful for learning

HTML elements are styled using CSS selectors and classes which references ID tags or element type tags. This groups all the CSS together and makes it easier to handle.

### 3. Style Sheet → BEST PRACTICE

Creating a separate .css file which is referenced in the HTML Doc and uses element ID values as references on where to apply style preferences. In this you don't need the opening and closing `style` tags.

CSS Uses a selector to target an HTML element in the DOM (Document Object Model) and then apply a variety of attributes to that element which changes the way it is displayed on the browser.

## Methods to Change Attributes of an Element

### Example: Colour of Text

For this example we will use inline styling to change the color of some text. We must add a `style` attribute to the opening tag to do this.

```
<h2 style="color:red;">This heading 2 is RED</h2>
```

**This heading 2 is RED**

Just content inside `h2` is changed

### CSS Selectors

These allow us to style multiple of the same element in a structured manner. This is done through declaring a set of `style` tags.

This can be done at the top or bottom of the code, but it mustn't be within any other element than the `html` tags.

### CSS Classes

This involves creating an entry in the style tags which is then declared in elements that need the specific styling properties.

```
<style>
  h2{
    color: blue;
  }

  .green-text{
    color: green;
  }
</style>
```

**This is Green**



```
<h2 class="green-text">This is Green</h2>
```

# CSS Properties

## Fonts

### Size

Font size is changed using the `font-size` property which takes a size in pixels `px` or other units which we will introduce later

### Colour

Color of text can be changed using the `color` property in the declaration. Adding the `!important` within the declaration ensures that this is the only styling color that h2 will use regardless of other colors it inherits.

### Font Family

A font family dictates the style of each character that will be used and is declared using `font-family` in the selector. It takes a Family name in single quotes

### Degrading a Font

Sometimes a font may not be available due to error or issue, we can add a backup font in case this happens. The `sans-serif` denotes the back up font, this is known as the generic name

```
<!-- This code is for examples -->
<style>
  h1{
    font-size: 30px;
  }
  <!-- h1 now has a fixed size -->

  h2{
    color: green !important;
  }
  <!-- h2 will always be green -->

  h3{
    <!-- Structure: FAMILY, GENERIC-->
    font-family: 'Hind';
  }

  <!-- All h3 will use Hind font -->
  p{
    font-family: 'Arial', sans-serif;
  }
</style>
```

## Import a Google Font

Google Fonts is a large directory of fonts to choose from and utilising them can bring a unique look to your page. In order to access the directory you need to use a `<link>` tag which is auto closing with `href` attribute for the URI, a `rel` attribute for the relationship and a `type` attribute to declare it as CSS. The link is generated for you on the Google Fonts website.

```
<link href="https://fonts.googleapis.com/css?family=Lobster"
rel="stylesheet" type="text/css">
<!-- This link makes the lobster font available to be used in the page -->
```

## Selectors

### Attribute Selectors

Attribute Selectors allow us to check if `[attribute = value]` then we are able to style elements with those specific values of the attribute. In the example it checks if the type attribute has a radio value, for radio buttons.

```
<style>
  [type='radio'] {
    color: blue;
  }
</style>
```

### ID Selectors

Id's can be assigned to individual elements, we can style the elements by using their IDs too. This requires the `#` to be placed before the id when declaring the styling. In this case the `id` is `option-button`

```
<style>
  #option-button {
    color: red;
  }
</style>
```

### Class Selectors

`class` is an attribute of HTML elements which can allow you to group and style elements of similar types together. Here, our class is called `primary-content` and is referred as a class using `.` at the beginning.

```
<style>
  .primary-content {
    font-size: 20px;
  }
</style>
```

# Content Styling

## Image Size

Image size is changed using the `width` property and setting it to a specific unit in pixels `px`, there are other units for size we will look. Images are styled using the `class` attribute in the `image` tags.

## Borders

Borders can be added to many elements, from text boxes, images and sections in a page. They are declared as a class and using various `border-` properties.

- Border-Radius makes rounded corners
- Can be a % unit , 50% makes a circle

## Background Colour

This is edited using the `background-color` property. It is added to an element through the `class` attribute.

## Padding

Padding is the space between the content of an element and its border. It is declared using the `padding` property with units in `px`.



Adding a direction after `padding-` such as `left`, `right`, `top`, `bottom` sets specific padding in that direction. This

```
<!-- This code is for examples -->
<style>
  .larger-image{
    width: 500px;
  }
  <!-- Length scales automatically -->

  .thick-blue-border{
    border-color: blue;
    border-width: 15px;
    border-style: solid;
    border-radius: 10%;
  }
  <!-- more customisation online -->

  .red-background{
    background-color:red;
  }
  <!-- Colours can be hex,rgb & rgba-->

  .blue-box {
    background-color: blue;
    color: #ffffff;
    padding: 20px;
  }
  <!-- 20px space inside box -->

  <!-- Clockwise Notation Example -->
  padding: 10px 20px 10px 20px
  <!-- top - right - bottom - left -->

  .blue-box {
    background-color: blue;
    color: #ffffff;
    padding: 20px;
    margin: 20px;
  }
  <!-- 20px space added outside -->

  <!-- Clockwise Notation Example -->
  margin: 10px 20px 10px 20px
  <!-- top - right - bottom - left -->

  .box{
```

can also be done using clockwise notation.

## Margin

Margin is the space between an element's border and surrounding elements. It is declared using `margin` property and takes units such as `px`. Adding negative margin makes the element larger. Adding a direction after `margin-` such as `left`, `right`, `top`, `bottom` sets specific padding in that direction. Clockwise notation can also be utilised with margins

```
--box-color: black;
}
<!-- box has only a variable in it-->
#box1{
  color: var(--box-color, gray);
}

:root{
  --var1-color: white;
  --var2-color: gray;
  --var3-color: yellow;
}
</style>
```

## CSS Variables

CSS variables are a very powerful way to change multiple properties at once by changing only one value. They are declared as properties with `--` before and can be referenced in other styling elements with a default value too.

## Roots

Adding a `:root` styling element will give the browser a default set of styling variables which it will use throughout and you can change when needed for specific sections. These are often declared at the top of a stylesheet.

## Size Units

### Absolute Units

Absolute units, such as `px` (pixels), tie to physical units of measurement like mm and in. They approximate the actual measurement on a screen and remain the

same across different screens. They become difficult to handle when you want your site to have varied structure if viewed on mobile.

## Relative Units

Relative units , such as `em` or `rem` , are relative to another length value. `em` is based on the size of the element's font. Relative units allow for responsive change in the elements when moving from screen size to screen size.

## Colour Units

### Hex Codes

Hexadecimal codes are used when you require very specific colours which do not fall in the general category of simple hues. They are often declared with 6 digits `#000000` . These 6 digits represent 3 components made of 2 digits each. Each component represents R, G and B. An Abbreviated Hex Code is one with 3 digits only `#000` , here each component has 1 digit only, limiting the number of colour possibilities from 16 million to a manageable 4000.

A hexadecimal digit can be 1 of 16 values, 0-9 then A-F.

### RGB

This is another way to set specific colors by using `rgb( 0, 0, 0)` . Each number can be set between 0-255 and represents R, G and B values. There is another extension to this, known as RGBA which also includes a transparency value from 0-1 in decimals, `rgba(100, 200, 30, 0.60)` for example.