Reference Guide

INTRODUCTION

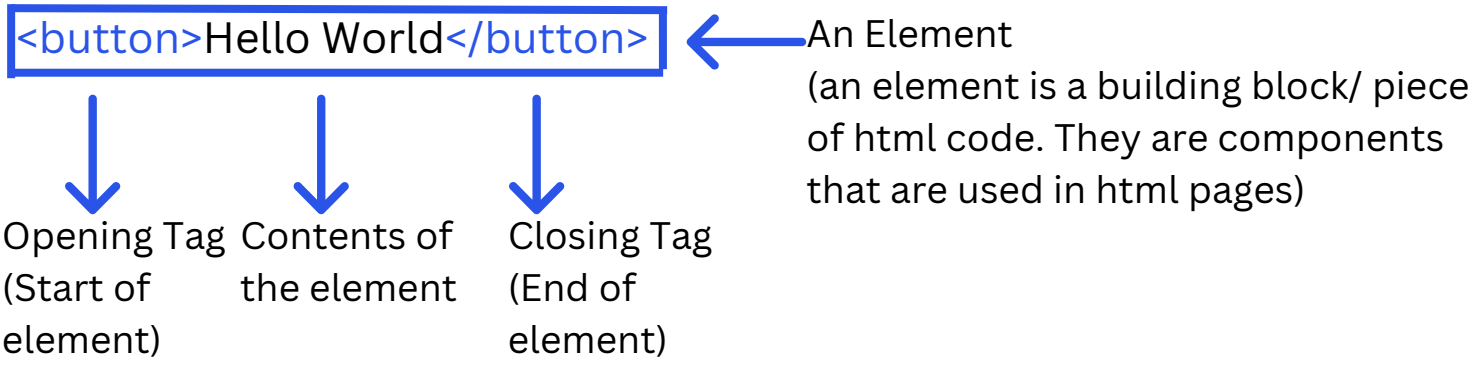
The Reference Guide gives you a helping hand when you're coding with HTML and CSS. It has an overview of everything you need to know as a beginner to intermediate HTML and CSS coder.

HTML BASICS

`<button>Hello World</button>`	Creates a button w/ the text "Hello world"
`<p>This is a paragraph of text</p>`	Creates text/a paragraph of text

Syntax Rules for HTML code:

1. Elements should have matching opening and closing tags.



2. Multiple spaces will always combine into only one space. All three of the examples below will show up the same on a webpage.

```
<p>paragraph of text<p>
```

```
<p>
  paragraph of text
<p>
```

```
<p> paragraph  of  text  <p>
```

Reference Guide

Attributes are an important part of an HTML element and can effect how it acts.

```
<a href="https://youtube.com">
  Youtube link
</a>
```

```
<a href="https://youtube.com" target="_blank">
  Youtube link
</a>
```

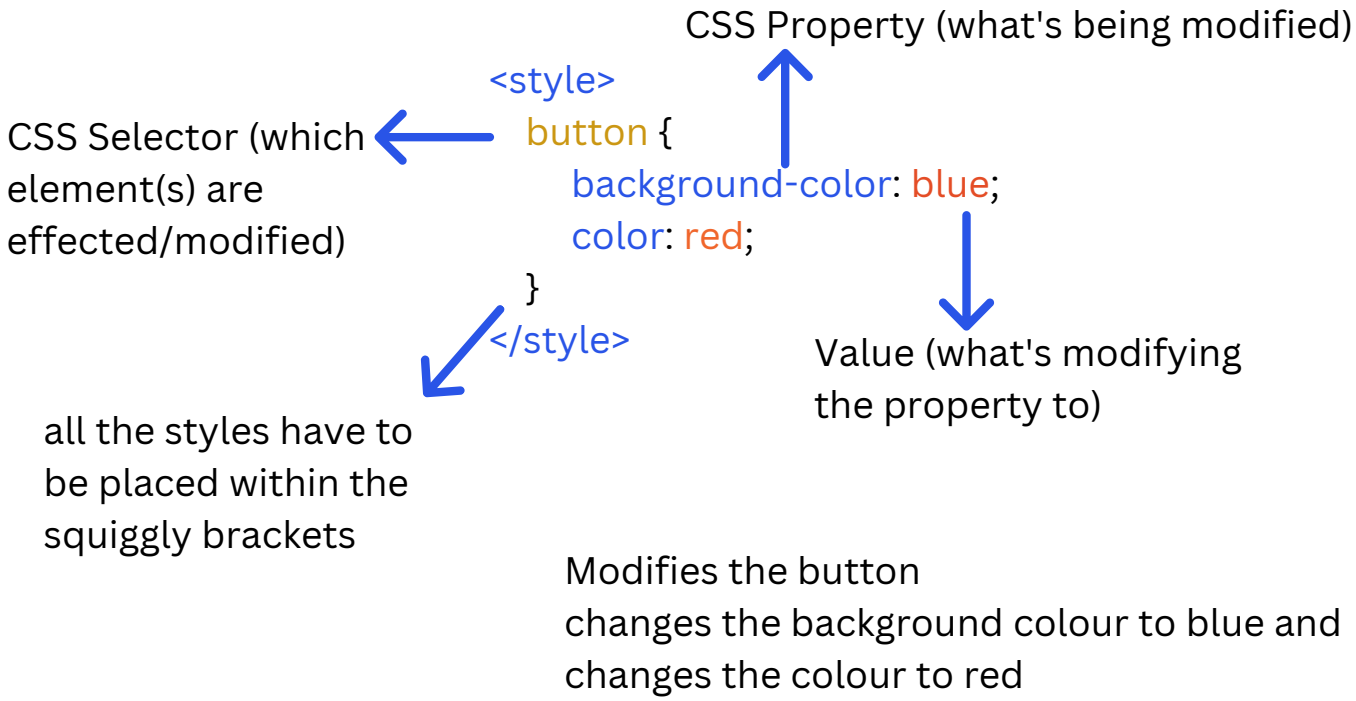
<a> lets you make a hyperlink or link to another website

href specifics the URL of the page that the link goes to

target lets you open the link in a new tab instead of redirection the page you're already one

CSS BASICS

There are multiple ways of writing CSS code, one way is using the <styles> HTML element.



Reference Guide

Here are some of the most common CSS properties and what they do/mean.

```
button {
  background-color: blue;
  color: red;

  height: 24px;
  width: 20px;

  cursor: pointer;

  border: none;
  border-radius: 5px;
  border-color: red;
  border-width: 5px;
  border-style: solid;
}
```

Sets the background colour to blue. (the value can be either a colour name(red), RGB (255, 105, 25), or hex (#0096FF).

Sets the text colour (takes the same values as the background colour does).

Sets the height (the value can be either a pixel value (24px) or a percentage (25%)).

Sets/changes the width (takes the same values as height).

Changes the mouse/cursor when hovering over the element.

Gets rid of the border.

Creates a rounded corner on the border.

Sets the border colour.

Sets the width of the border.

Sets the style of the border (the common values are solid, dotted, dashed)

Most people use Google to search for specific CSS properties. If you don't know or don't remember a CSS property, you can always use Google and search up what you want to do with the CSS property and you'll find what the property is.

ex. "CSS text underline", "CSS rounded corners", "CSS solid border"

Every CSS property has set values that are allowed (color allows colour values, height allows measurement values, etc) Here is a list of some value categories that are very useful to know.

Reference Guide

COLOUR VALUES:

1. colour name - red, blue, green
2. RGB - rgb(0, 105, 25);

A more specific way to measure colour. Each value in the brackets represents the amount of red, green and blue are used. (0=red, 105=green, 25=blue). There is a max value (255) and a minimum value(0) for each colour. Black in RGB is (0, 0, 0) and white is (255, 255, 255).

3. Hex value - #0096FF

Each character is base 16 (each can have a value of 0,1,2,3,4,5,6,7,8,9,A(10),B(11), C(12), D(13), E(14),F(15) (16 possibilities)). The first 2 characters represent red, the second two are green, and the last two are blue. The amount the two numbers/letters add up to is the value of the red, green, or blue colour in the final colour.

4. RGBA value - rgba(0, 105, 25, 0.5)

This works the same as RGB but the fourth value (a) determines the opacity of the colour. 0= completely see-through, 0.5 is 50% see-through, 1 = 100% solid colour

MEASUREMENT VALUES:

1. Pixels = 50px, 100px

Pixels (or px) are the most common type of measurement used in CSS. They are used a lot in the digital world, for example, a 4k screen is 3840px by 2160px.

2. percentage - 50%, 100%

Certain CSS properties like width: 50%; use percentages. And that means that 50% of the original width or height of the element.

Class Attributes let us target specific elements using CSS.

```
<button class="home-button">
   Home
</button>

.home-button {
   ...
}
```

how to write a class in your element

how you target a class using the CSS selectors

You can add a class to an element by adding in class="class-name" in between the <button and the >. This allows you to target all elements on the page with that specific class name in your CSS code. (Multiple elements can have the same class, and elements can have multiple classes).

Reference Guide

Intermediate CSS Properties

```
.button {
  opacity: 0.5;
```

You can set how see-through an element is
0.5 = 50% see-through, 0=invisible, 1 = not see-through (this is the default value).

```
  transition: <property> <duration>;
```

Transition smoothly when changing styles (typically used with hovering). You can transition multiple properties by separating them with a comma.

```
  box-shadow: <h-position> <v-position> <blur> <color>;
}
```

Creates a shadow around (either to the right, left, top, or bottom) the element and sets the colour of the shadow.

Chrome Dev Tools

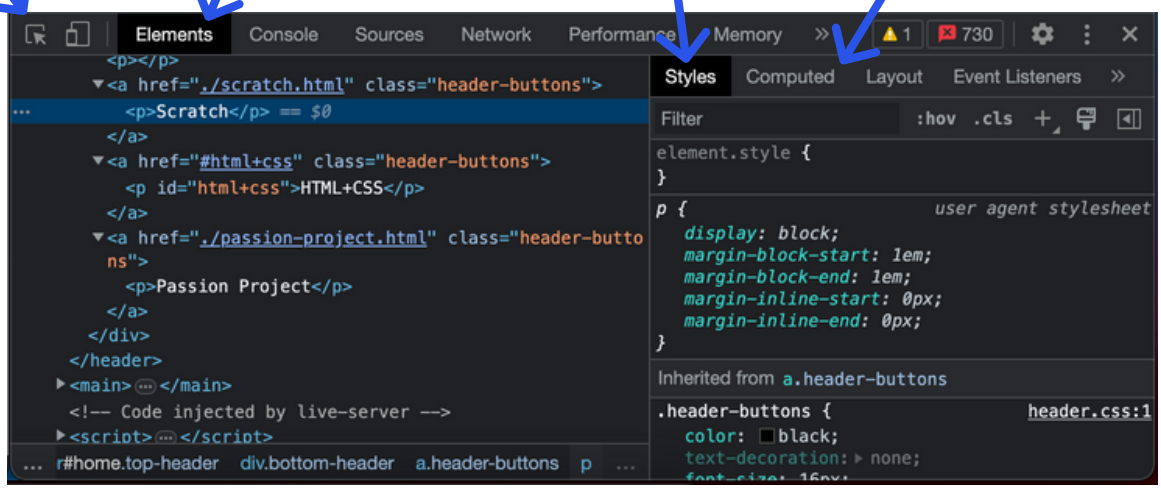
Chrome Dev Tools let us view and modify the HTML and CSS of a website directly in the browser, to access and open DevTools you can right click on any part of the screen and click inspect.

Find an element by hovering/clicking on it

View the HTML

View and modify the CSS

View the CSS final computed values



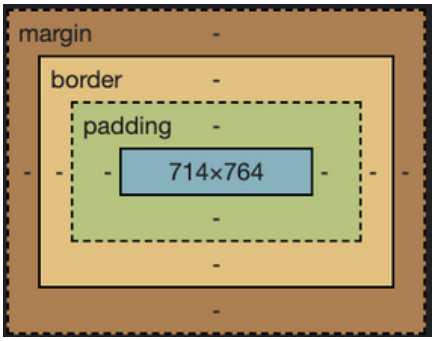
Reference Guide

Comments

Comments are very useful and let us write code that the browser will ignore. It's useful to write notes about what your code does or how it works. You can also comment out parts of your code if you don't want the browser to apply and read the code but you also don't want to delete it.

```
<!-- This is a HTML comment -->
/* This is a CSS comment */
```

CSS Box Model



Margin = the space on the outside of the element
Padding = the space on the inside of the element
Border = the border around the element

Certain elements like <p> have a default set margins so If you want no margin, you would have to set it to 0.

```
.button {
  margin: 10px;
  margin-left: 10px;
```

This adds 10px of space on the outside of the element on all sides. When there is a -left (this can also be -right, -top, -bottom), then it adds 10px of space on the outside of only on the specific side of the element.

```
  margin: <top&bottom> <left &right>;
  margin: <top> <right> <bottom> <left>;
```

Theses are some shorthand versions of the element.

```
  padding: 10px;
  padding-right: 10px;
```

This adds 10px of space on the inside of the element on all sides. When there is a -right (this can also be -left, -top, -bottom), then it adds 10px of space on the outside of only on the specific side of the element.

```
  padding: <top&bottom> <left&right>;
  padding: <top> <right> <bottom> <left>;
```

These are some shorthand versions of the element.

```
  border-style: solid;
  border-color: blue;
  border-width: 1px;
```

These CSS styles allow you to edit the border of the element. You can change the colour, width, and style (solid, dotted, dashed, etc) of the border.

```
  border: <width> <style> <color>;
```

This is a shorthand version of the element.

```
}
```

Reference Guide

Text Styles

```
.button {
  font-family: Arial;
  font-size: 30px;
  font-weight: bold;
  font-style: italic;
  text-align: center;
  line-height: 24px;
  text-decoration: underline;
}
```

This lets you change or set the font, if you add ", second-font-type" after writing the first font, that lets you set a backup font of your choice.

Lets you change the text size.

Lets you change the thickness of the text. You can also set a pixel value to change the thickness.

Lets you make the text italic.

You can align your text. Other values you can use are justified, left, and right.

Lets you change or set the space between text lines.

Underlines the text. You can set it to other values like none and overline.

Certain text elements like the <a>, <u>, , and can appear within a line of text or within a <p> element and you can style theses elements using a class. It's useful if you want to style only a specific part of your text.

```
<p>
  This is a <strong> text element</strong>
</p>

<p>
 This is a <strong class="strong-text"> text element</strong>
</p>
```

Reference Guide

Images

To display images, you first need to allow your HTML code to have access to the photo. Typically, people will create a new folder, within your code folder, called images and then save the photo into that folder.

```
<img src="images/photo.png">
```

You use this element to add in images, you first write the name of the folder you are keeping your photos in and then write a "/" and then write the name of your photo. Image elements are a bit different as they only have one tag so you don't need any closing tag to write the element.

You can use a class to style the photo as well.

```
.image {
  Width: 200px;
  height: 100px;

  object-fit: contain;

  object-position: left;
}
```

This sets the image width.
This sets the image height.
If both the width and the height are set, the image may stretch.

This lets you choose how the image appears within a set dimension, contain shrinks the image so it's contained in the width and height area. Cover, another value for object fit, makes the image bigger to cover the whole width and height area without stretching or distorting the photo.

This lets you determine where the image is positioned within the width and height area, left, right, top and bottom are all the values you can use.

Reference Guide

Inputs

```
<input type="text">
```
This creates a text box

```
<input type="checkbox">
```
This creates a checkbox

```
<input type="text" placeholder="Search">
```
Adding a placeholder will label to the text box to "Search" until its clicked on and typed in.

```
.input-text {
  font-size: 30px;
}
```
You can also add classes to target the element as usual, and you can use specific CSS styles like the font-size style, for example, to change its appearance.

```
.input-text:placeholder {
  font-size: 30px;
}
```
If you want to change the placeholder text, you would use the ":placeholder" when targeting the elements class to style it.

HTML STRUCTURE (This is code you need to have for your webpage work)

```
<!DOCTYPE html>
```
Tells the browser to use a modern version of HTML

```
<html>
  <head><head>
  <body><body>
<html>
```
<head> containing everything that's not visible, like the title and if you want to link fonts or CSS stylesheets.

<body> contains all the visible code like buttons, text, images, etc...

Inside of the <head> element, you would put:

```
<title>Title in the tab</title>
```
This lets you set the title in the tabs

```
<link rel="stylesheet" href=" URL here">
```
This lets you link a URL to a font, for example, from Google, onto the page.

```
<link rel="stylesheet" href="folder1/styles.css">
```
This lets you use file paths to link your CSS page to your HTML page.

There are other ways to write it. The "/" mean it will go inside the folder written.

```
href="styles.css">
href="folder1/folder2/styles.css">
```

Reference Guide

CSS Display Property

```
.element {
  display: block;
  display: inline-block;
  vertical-align: middle;
  display: inline;
}
```

The element will take up the whole line in its container.

The element will only take up the space it needs.

This sets the vertical alignment of an inline-block element.

Elements will appear within a line of text (text element).

Element

This element is a container. You can put other elements within it to group them together (including other s, which is called nesting). You can give elements classes just as you would for any other element.

```
<div class="container">
  <button>Yes</button>
  <button>No</button>
</div>
```

This is a simple example of how an element is used to group other elements together. You can use CSS styles for the element just as you would for any other element, except the styles affect anything within the as one element together.

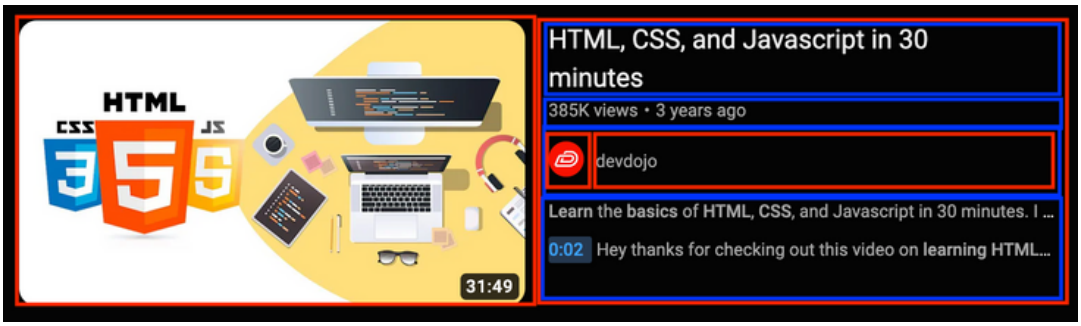
Semantic Elements

These elements work the same as , but they also give the HTML meaning when things like screen readers, search engines, and other devices read the website. The most commonly used semantic elements are <header>, <nav>, <main>, and <section>.

Nested Layouts

Nested Layouts let you create either vertical or horizontal layouts using elements. You can make more complex layouts by putting vertical layouts inside a horizontal layout or vice versa. Here is an example of a more complex layout from a popular website (Youtube).

Reference Guide



Blue represents the Vertical Layouts

Red represents the Horizontal layouts

To create a vertical layout, you use s with display: block (this is more common) and use a flexbox (explained later) with flex-direction: column. Then you use a CSS grid (also explained later) with 1 column.

To create a horizontal layout, you use s with display: inline-block (less common/not recommended) and use a flexbox with flex-direction: row. Then use a CSS grid with multiple columns.

Inline CSS Styles

This is another way to write your CSS styles. You would use style="..." and put it in the same place as where you would write the class.

```
<div style ="
  background-color: white;
  color: red;
">...</div>
```

inline style is the CSS written within the line of HTML and they only affect the element with the style=".." attribute.

CSS Grid

```
.grid {
  display: grid;
  column-gap: 20px;
  row-gap: 40px;
  align-items: center;
  justify-content: center;
}
```

This turns the element into a grid container
This sets the space between columns.
This sets the space between rows.
Aligns the columns vertically in the center.
This aligns the columns horizontally in the center, you can use space-between as another way to justify content and spread out the columns evenly horizontally.

Reference Guide


```
.grid {
  grid-template-columns: 100px 100px;
```
Sets how many columns are in the grid and how wide they are.

```
  grid-tempalte-columns: 100px 1fr;
```
1fr means the column will take up the remaining space in the grid container.

```
  grid-tempalte-columns: 1fr 1fr;
```
If all your column takes up a space of 1fr, they will take up an equal amount of the remaining space.

```
  grid-tempalte-columns: 1fr 2fr;
}
```
The number in front of the "fr" is how much space the column gets, so if you have "1fr 2fr" the second column will take up twice as much space as the first column.

Flexbox

A flexbox is another way of creating a grid, but it is more flexible and you can make it more complex with how your elements are arranged.

Creates a flexbox

```
.flexbox {
  display: flex;
```
This sets whether the flexbox has rows/horizontally or has columns/vertically (flex-direction: column;) default, it's set to be in rows.

```
  flex-direction: row;
  justify-content: center;
```
This centers the elements in the flexbox horizontally (if set to rows) and vertically (if set to columns). You can also set it to space between which spreads out the elements evenly across the horizontal space (if set to row) or vertical space (if set to columns)

```
  align-items: center;
}
```
The centers the elements in the flexbox vertically (if set to rows) and horizontally (if set to columns). You can also use space-between, which spreads out the elements evenly across the vertical space (if set to row) or horizontal space (if set to column).

Reference Guide

If you are using inline CSS styles, this is how you would create a flexbox. Each element represents a row.

```
<div style="
  display: flex;
  flex-direction: row;
">
```

creates the flexbox
and sets the flexbox to rows

```
  <p style="flex: 1;">
    element 1
  </p>
```

The element takes up 1/3 of the remaining space.

```
  <p style="flex: 2;">
    element 2
  </p>
```

The element takes up 2/3 of the remaining space.

```
  <p style="width: 100px;">
    element 3
  </p>
</div>
```

The element takes up/has a width of 100px.

```
.element-within-the-flexbox {
 width: 100px;
```

Sets the width of the flexbox element (sets to 100px)

```
 flex: 1;
```

Takes up the remaining amount of space. the number/value determines how much space.

```
 flex-shrink: 0;
```

Dosent allows the element to shrink when resizing the screen.

```
 width: 0;
}
```

Allows the element to shrink down when resizing the screen.

Reference Guide

CSS Position

CSS Position lets you create elements that can stick to the page while scrolling and appear on top of other elements. This is helpful for creating things like headers of webpages.

```
.element {
  position: static;
```
This is the default for every element, and it makes the element display normally.

```
  position: fixed;
```
This lets the element stick to the page while scrolling (positions it in the browser window).

```
  position: absolute;
```
This lets the element scroll along with the page but won't stick while scrolling (positions it on the page)

```
  position: relative;
```
This lets the element appear normally (like if its position: static) but it can be pushed around with the top/bottom/left/right.

```
  left: 50px;
```
Sets how many pixels away from the left of the browser window the element is.

```
  right: 100px;
```
Sets how many pixels away from the right side of the browser window the element is.

```
  top: 0px;
```
Sets how many pixels away from the bottom of the browser the element is.

```
  bottom: 10px;
```
Sets how many pixels away from the top of the browser window the element is.

```
  left: -5px;
```
If you use negative pixels the element will be placed beyond the selected edge of the screen/web page.

For position relative (with top/bottom/left/right), it will push the element however many pixels away from this original position to the selected side.

```
  width: 100px;
  height: 100px;
}
```
Sets the element's width.
Sets the element's height.

Reference Guide

When you have a "position: __" element inside of another "position: __" element, if they are set to different position values, they will apply the one from the outer element.

Z-Index lets you determine which elements appear in front or behind the others. You can set the element's z-index, and whichever has the higher z-index will appear in front of the other elements. The lowest a z-index can be is 0.
If two elements have the same z-index, the one written later in your code will appear in front.

```
.element-1 {
   position: absolute;
   z-index: 1;
}
```

```
.element-2 {
   position: fixed;
   z-index: 2;
}
```

Element 2 will appear in front of element 1 because it has a higher z-index.

Responsive Design

This makes your webpage look good on any screen size.

```
@media (max-width: 250px) {
  .element {
     width: 350px;
  }
}
```

Only applies the CSS code when the screen has a width between 0px and 250 px.

```
@media (min-width: 250px) and (max-width: 500px){
   .element {
      width: 350px;
   }
}
```

Only applies the CSS code when the screen has a width between 250px and 500 px.

```
@media (min-width: 500px) {
  .element {
    width: 600px;.
  }
}
```

Only applies the CSS code when the screen has a width greater than 500px;

Reference Guide

Advanced CSS Selectors

There are lots of ways to select multiple specific elements with CSS Selectors. You can use commas or spaces to select multiple at the same time.

With spaces:
`.class1 img { ... }`

Targets the <img> elements that are within the targeted class

`.class2:hover .tooltip { ... }`

Targets the .tooltip but only when hovering over the elements within the targeted class.

With Commas:
`.class 1, .class 2 { ... }`

Targets multiple classes at the same time

`.class1, p { ... }`

Targets all the <p> elements at the same time as the targeted class

With spaces and commas:
```
.class1 img,
.class2 .tooltip { ... }
```

Targets the <img> elements within the first targeted class and targets the .tooltip within the second targeted class

Inheritance

When text properties are set in an outer element, they are passed down into the inner elements. You can set styles you want on the entire page within the <body>, and it will affect all of your code within and can reduce the repetition of CSS code. These re both examples of inheritance

```
<div style="color: red;">
   <p>Text</p>
</div>
```

the colour of the text will be red

```
body {
  font-family: Arial;
  color: red;
  font-size: 24px;
}
```

Every element within the <body> will use those css styles by defualt

Reference Guide

CSS Specificity

If you have multiple CSS Selectors changing the same properties for one element, CSS Specificity helps decide which selector gets applied.

Here are a few rules (a trick to know is that the more specific it is, the higher priority it has):

1. Inline CSS has higher priority than .class selectors
2. .class selectors have higher priority than element name selectors
3. Element name selectors have higher priority than inheritance from <body>
4. If two selectors have the same priority the one written alter in the code is applied

Other CSS Properties

These are a few other CSS Properties that are helpful to know

```
.tooltip {
  pointer-events: none;
  white-space: nowrap;
}
```

Turns off all interactions (clicks and hovers) with the mouse.

Stops the text inside an element from wrapping to multiple lines.