A Cheatsheet Guide to understanding **CSS Basics**!

What is **CSS**?

CSS is an acronym for **C**ascading **S**tyle **S**heets. CSS is made of what we call **style rules**, or simply **styles**.

What do we use these **style rules** for?

Style rules are ways we can alter/affect the *presentation* of content on a web page. In other words, styles affect how the layout of the page and its content looks to the user. Good design presentation can make or break a web page.

Ok, so what does a **style rule look like**?

Each style rule will have *three parts* to it which you will code. They are the:

1. **Selector**
   A selector is a string you give to tell the browser which HTML tag(s) it should select to apply this style rule to.

2. **CSS Attribute name**
   CSS has many attribute names that describe what you presentation changes you would like to apply to the content (or box) of the selected tag(s).

3. **CSS Attribute value**
   Each attribute name will have an associated value (or set of values) that tell the browser exactly how it should change the content (or box) of the selected tag(s).

*Note that a style rule will only have one selector, but can consist of one or more CSS attribute name: value pairs.

Did you notice the comment above about **attribute: value pairs** above? *The CSS attribute names and values will always be coded in pairs with the attribute name followed by a colon and then the attribute's value*.

We'll cover exactly what coding a style rule looks like in just a bit, but first we have to understand the three types of style rules…. Let's talk about those next!

There are *three ways CSS Style rules can be applied to a web page*:

1. **Inline** styles

- are coded directly inside HTML tags as an HTML tag attribute named **style**
- this type of style rule only selects and affects the tag it is an attribute of => most specific style

Example:
Let's style a particular paragraph (<p>) tag in our web page such that its content text is blue in color and its font size is resized to 14 pixels. Using an inline style attribute (which I've bolded below) allows us to apply these two style attributes to just the content of this one tag.

*There is no separate selector needed for inline styles since the tag their style attribute is inside of is the selected tag whose content we'll apply the style attributes to.

Note there are two style *attribute: value pairs* here: **color: blue;** and **font-size: 14px;** that are being applied to the paragraph tag.

```
<p style="color: blue; font-size: 14px;">
Four score and twenty years ago...
</p>
```

2. **Embedded** (global) styles

- are coded inside a **<style>** tag which MUST ALWAYS be inside the **<head>** tag
- this type of style is applied only to selected tags in the .html file that the **<style>** tag is in

*We'll see an example of these below after discussing external styles as embedded and external styles are code the same way. The only difference between them is where they are coded.

3. **External** styles

- are coded inside a separate file ending with a **.css** extension
- this external (separate) .css file can then be linked (included) into *any .html file* we wish using the **<link>** tag (note that the **<link>** tag should also ALWAYS be inside the **<head>** tag)
- this type of style is the most flexible because the .css file that includes the styles can be included in as many .html files as we wish – yay!

What a style rule looks like for **external** and **embedded** styles:

```
selector {
  CSSAttributeName: value;
  CSSAttributeName: value;
  .
  .
  .
  CSSAttributeName: value;
}
```

The **selector** is used to select the HTML tag(s) that you want to apply the styles to...

The **attribute: value;** pair(s) specify what changes should be made to the presentation of the content of the selected tag(s).

Do you see the three parts of a style rule we mentioned earlier? 1) the selector, 2) CSS attribute name(s), and 3) attribute value(s).

Also, note the specific syntax shown above:

- The selector is followed by a pair of curly braces, {}'s, (required) which are usually best placed as shown above
- Inside the {}'s, we place the CSS attribute name: value; pair(s) which are ALWAYS ended with a semicolon (;)
- We also generally place each name: attribute; pair on its own line and indent it two spaces for better code readability

The only difference between external and embedded styles is where they are coded.

- **Embedded styles** are coded in a single .html file inside a **<style>** tag which must be inside the **<head>** tag *AND only select tags that are coded inside that same .html file*.

- **External styles** are coded inside a separate file ending in a **.css** extension. *The .css file can be linked into any .html file* using the **<link>** tag inside the **<head>** tag in the .html file.

What kind of **selectors** can we use? Ok, let's start with the most common…

- Using an asterisk (*****) as your selector will select every tag in your .html file

- Select **by tag name** (without the <>'s) which will cause *all tags of this name in the web page to be selected*:

  Example: **h1** {font-size: 10px;}

  this example selects all <h1> tags on this web page and applies a font size of 10 pixels to each <h1> tag's text content.

  Note that multiple attribute: value; pairs can be put on the same line, but that is not considered good coding practice -> bad code readability. However, if only one attribute: value; pair is being coded for the style rule, then putting it all on one line is considered acceptable. You can certainly put the {}'s and attribute: value; pair on their own lines as well.

- Select **by id** (using the **#** prefix) This is the *most specific way to select a tag*.

  id attribute values must be *unique* in a .html file which means only one tag on the page can have a unique id attribute value. Most (and multiple) tags can use the id attribute, but the values of the id attribute must be unique on the page.

  Example:

  HTML:
  ```
  <p id="stuff">Some paragraph text here</p>
  ```

  CSS:
  ```
  #stuff {
    font-family: Arial;
    color: green;
  }
  ```

  So, here, we have an HTML <p> tag with an id attribute with a value of "stuff". To apply a CSS style rule to this specific tag only, we select the tag by its id attribute value using **#stuff** as the **selector**. We change the paragraph's text font to Arial and change the color of the text to green. No other tag content is affected on the page by this style.

- Select **by class** (using the **.** prefix)

  The **class** attribute can be used in most HTML tags and more than one tag's class attribute can have the same class name value. When tags are selected by class in CSS (using the period), the styles in that style rule will be applied to every tag that has that class value.

  By the way, an HTML tag's class attribute can be assigned more than one class name value. They just need to be separated by a space inside the double quotes.

Example:

HTML:

```
<p class="abc">some paragraph text here</p>

<div class="abc">
  <a href="http://www.google.com">Google</a>
  Some text in the div tag…
</div>
```

Now our CSS:

```
.abc {
  font-family: Arial;
  color: green;
}

p.abc {    /* now, get more specific */
  color: blue;
}
```

The **.abc** selector selects any tag that has a class attribute with a value of "abc" and applies an Arial font with green color to that tag's text content.

The **p.abc** selector can be used to select only <p> tags that have a class attribute with a value of "abc" and turn the selected paragraph(s)' text blue. Placing a tag name in front of the class selector (the period) allows us to select only those types of tags that have this class name.

Note that **comments** in CSS code are begun with **/*** and ended with ***/**. These are block comments that can take up part of a line, an entire line, or multiple lines and everything between them is ignored by the browser. Comments are intended for humans only.

- Select **by descendant** (using a single *blank space* between two or more parts of our selector) where the descendant tags can be any level of descendant like a child, grandchild, great-grandchild, etc…, or select **by direct children** descendants using the > symbol between parts of a selector.

  In CSS, a **space** inside a selector is called a **descendant selector** which means that the browser will look for this tag *at some level inside a parent tag*.

  A **>** inside a selector is called a **direct child selector** meaning the browser will only select tag(s) that are *direct children of the parent tag on the left side of the* ***>.*** This selector actually runs more efficiently than the blank space descendant selector so it should be used when possible.

  Example:

  HTML:

```
<div class="abc">
   <div>
      <p>Some paragraph text</p>

      <div>
         <div id="someID">some div text</div>
      </div>
   </div>
</div>
```

  Wow, that's a lot of div's!

  CSS:

```
.abc div { /* note the blank space between .abc and div */
  color: red;
}

.abc p {
  color: green;
}

.abc > div {
  color: purple;
}
```

  We should read (and think about) selectors from **right-to-left**. With a descendant selector, the item to the right of the space will be a descendant of the item on the left of the space.

So, the descendant selector **.abc div** would read from right-to-left:

Select all <div> tags that are descendants of (inside of) any tag that has a class value of "abc" (in our case our first <div> tag above.

This means that the div#someID's "some div text" will turn red because it is one of the descendant <div> tags that have been selected.

What about the paragraph's "some paragraph text"? It will be green because the selector **.abc p** above selected all <p> tags that are descendants of our div.abc tag.

*Note that if we removed the **.abc p** style, then the paragraph's text would turn red as it is inside one of the previously styled <div> tags with the red color applied to any text content.

This reflects how CSS handles **inheritance**. CSS attribute changes are automatically passed down to descendants. Descendants can also then reset those values if you don't want the specific value(s) inherited (which is what we did above with the **.abc p** style).

For the above selector, **.abc > div**, only the first <div> tag inside <div class="abc"> will be selected as it is the only <div> that is a direct child of the div.abc tag. We would describe it as "select any <div> tags that are direct children only of any tag that has a class of "abc"".

*By the way, notice many web developers use the notation **div#stuff** to reflect a <div> tag that has an **id attribute** set to a value of "**stuff**" while the notation **div.abc** would reflect a <div> tag that has a **class attribute** set to a value of "**abc**".

- Using **multiple selectors** in the same style rule using **comma(s)** to separate them…

You can combine selectors to apply styles to more tags at the same time using commas. You can use multiple commas as well.

Example:

CSS:
```
li, a {
  color: green;
}
```

This example will select all <li> tags and all <a> tags in the page at the same time and apply the color green to their text content.

So, what does the Cascading part of CSS refer to exactly?

First of all, it has to do with the three types of styles we learned about above. Each style type is a specific part of the cascade.

The **cascade** in Cascading Style Sheets *refers to the order* in which multiple, and potentially conflicting style rules are applied to web page tag content.

The **Cascade** for precedence (overriding order meaning which style takes precedence over, or wins, vs another style) in CSS is:

> Inline styles override Embedded styles which in turn override External styles. This means that Inline styles will always win (override) when conflicting with any of the other two types of styles. They are the most specific (pinpoint power) style and give you, the developer, absolute control of any specific tag.
>
> Thus: **Inline > Embedded > External**
>
> However, that doesn't mean you always use **inline** styles. In fact, they are also the least flexible of the style types as they can only apply to an individual tag's content at a time.
>
> The most commonly used style type (by far) are **external** styles, even though they get overridden whenever they are in conflict with either of the other two style types (*inline* or *embedded*) on a specific tag.
>
> They are easily the most flexible type of style since *they are located in a separate .css file which can be included into any (and multiple) .html files* so the styles in an external .css file can affect many web pages at the same time.
>
> Thus, the **most flexible to least flexible** order of styles is: **External > Embedded > Inline**

CSS Reference Websites

- MDN CSS website – MDN (Mozilla Developer Network) is the Firefox browser folks and they have put together a CSS Reference, tutorials, and guides website that is one of the best!

- Codrops CSS reference – www.codrops.com is an amazing website that shows experimental effects on web pages using CSS and JavaScript. All of the code is available for free and there are also lots of great web design and development related resources discussed and made available.

- CSS Tricks website – This website has all kinds of articles and guides with great examples on every topic imaginable related to CSS and HTML. Check out their guides.