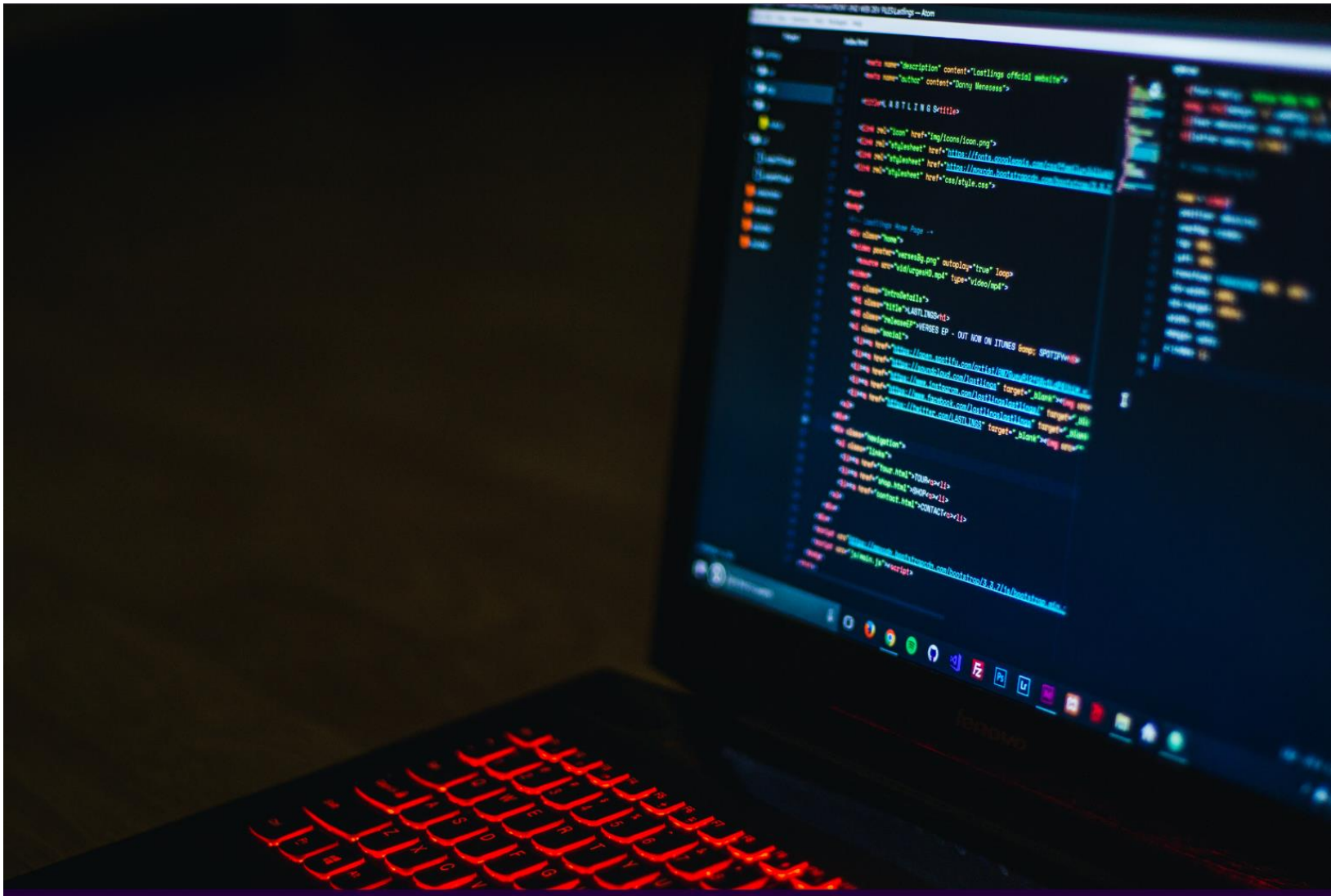
Computer Coding

Learn to Program



Intro To CSS

Astro Clare Technology

Astro Clare Technology © Flint, MI EST 2022

“ HELLO CODER “

We at Astro Clare Technology are proud to have you receive this book! There will be many fun activities for you to enjoy. If you are a parent purchasing this book for your child to venture into they great world of becoming an software engineer, we thank you for trusting us ! We have many great activities for you and your child to enjoy together.

Our main objective is to get individuals to begin their journey in coding, programming and computing technologies. We see the lack of resources decline over the years and understand that purchasing these courses at Universities are even more challenging financially and timely. So we will due more than just “teach” you how to become the greatest engineer!

You are going to learn many skills such as: critical thinking, logical outlooks, problem solving skills, discipline and career ready structure and even more !

Each successful assessment from you will grant an Certificate from Astro Clare Technology along with an PDF of your course completion to show your next employer.

Once again, we thank you for trusting in Astro Clare Technology and hope you put your best code forward.

Clarence Scott
CEO & Founder

UNVEILING CSS

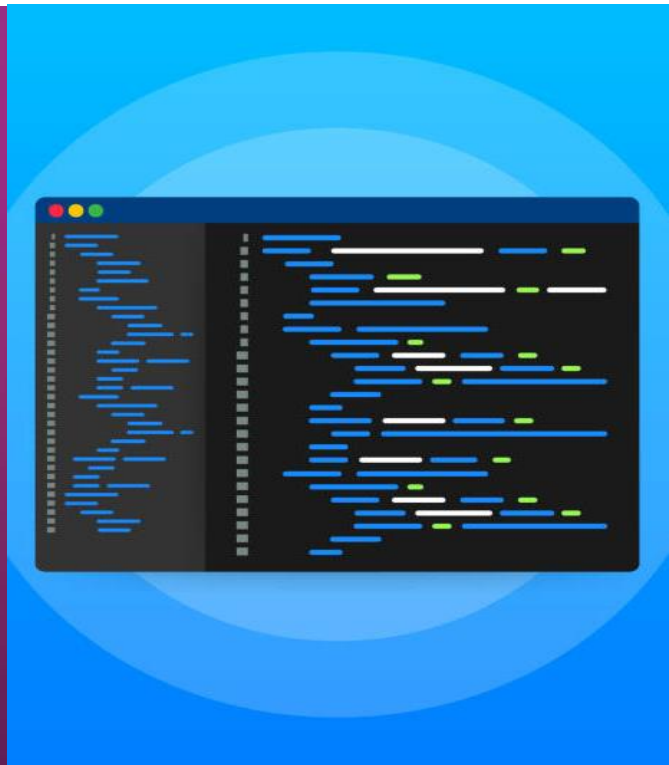
Actual name for CSS is, Cascading Style Sheets. CSS is used to change colors , font size, etc., as well as created layouts. Layouts are useful for navigation menus, columns, sidebars and many more things.

If you remember from our HTML tutorials, we used elements to create layouts on our HTML page. CSS uses ***rules*** to style those elements. An rule can cause all div containers to have a blue background with white text. A rule is a group of style ***declarations.***

There are many ways to apply cascading styling sheet, inline, in HTML using ***<style> </style>,*** or by adding the CSS document from an external sheet. Using the **style** attribute on HTML elements applies the style to that element only. We would have to copy the same style attribute and define it on all the elements needed. To avoid redundancy, we want to avoid doing this. So what can we do??

It is common practice to define CSS rules in a different document and link it to our HTML document. You can target elements ***selectors,*** this allows you to target specific elements and apply the same styling.

To add an external CSS document to your HTML page, simply add:
< link rel="stylesheet" href="style.css"> be sure to have an ***style.css*** document created in your projects folder.



RULES & SELECTORS

It's best to learn about Rules & Selectors prior to using CSS sheets. The style rules are defined in a CSS file using **selectors,** these are your elements from your HTML documents.

header {color: green;}

Here we are targeting all of the header elements of our HTML document and changing the text color to green.

selector { property: value; }. Declarations are placed inside of curly braces. We also can add multiple styles for our selector, example:

header { background-color: green; color:red; }

Multiple selectors can be styled at once by using a comma to separate them. This will make all of the selectors have the same **property** and **value.**

selector1, selector2 { property: value; }

CLASS & ID SELECTORS

The **class selector** is very powerful when we are styling our HTML documents. Often there may be a single element or just a set of elements you need to style, without changing the others. This is where we will use the **class selector,** we will need to add a ***class*** attribute to the elements we will be targeting.

<h1 class="heading">CSS is awesome</h1>
** CSS Intro **

After you've done this, we can make the class selector in CSS. All class selectors needs to with a **dot,** followed by the name we gave to our element.
.heading { color: green; background-color:gray; }

Anything we decide to add this class selector too will have the same styling, simply by adding the attribute.

To create an **id** selector, you simply need to use the hash symbol, **or hashtag** #, before the value rather than the **dot**. Only one element can have the same **id** on the page.

<h1 class="heading">CSS is awesome</h1>
** CSS Intro **
#firstP { color: green; font-weight: bold; }

We can also target location based elements. For example if you have an element **<b>** inside of a list **<li>**, you can target that element using:

li b { color:orange; }

Styles that are given to parent elements are inherited by the child elements.
If you decide to change the **color** or **font** on an element every element inside of it will also be styled.

TEXT STYLING

Styling text is a way to capture the eye of your web page visitors. In the previous page we have already seen how to change text color with the property color. As we learned in the HTML portion of this book we can use: color names, hex & rgb to change color values in our styling. Along with this we have many ways to change our **font-size**

This property has many other values that can change the sizing but we will focus on using pixels(px). Load up a generic page for this exercise and type up a single paragraph, we will change the styling of the paragraph in our document to a **font-size** of **48px**. This is definitely over and beyond for what we are trying to accomplish but I want you to have something that sticks out to you.

Sometimes it is best to change the font-family of the text throughout your webpage. There are a group of web safe fonts that include: Arial, Times New Roman, Verdana and quite a few more. In order for a font to work in needs to be available on your device in downloaded into a folder for the web page to generate the proper font.

The best thing to do when styling your font-family is to chose multiple fonts so that if one is not available the browser will go to the next available font. We can also use the font-style property to make text italic, font-weight is used for the boldness of your text. 100 (thin) to 900 (thick). 400 is normal text thickness while 700 is the equivalent to bold.

Using the text-transform allows you to change text in regards to: uppercase, lowercase, capitalize:

- **uppercase: Transforms the text to capitals.**
- **lowercase: Transforms the text to lower case.**
- **capitalize: Transforms all words to have the first letter capitalized.**

Text-decoration allows you to: underline, overline or line-though text.
text-shadow: (x-axis) (y-axis) (radius of shadow) (color);

Example: .**heading { text-shadow: 2px 3px 5px green; }**

Multiple shadows can be developed by comma separation.

Example: **.heading { text-shadow: 2px 3px 5px green, 6px 8px 2px red; }**

Aligning your text gives you control over the container text layout. Values can either be aligned by the text-align with: left, right, center, justify. Sometimes you may need additional line separation. This can be achieved by using lineheight property. Simply inputting line-height: 1.4; will cause the line to become 1.4 times the height of the current font. Letter and word spacing can cause desired spacing for clarity using certain fonts. Simply use letterspacing or word-spacing and chose the pixels that you want apart from your letters or words. Example: header{ word-spacing: 8px; or letterspacing: 3px; }

BOX MODEL

The box model is something you should keep a mental visualization of. This helps you maintain organization on your web page when styling different elements.

The **content** box is the area where the content is displayed.

The **padding** box is the teal space around the content.

The **border** box comes after the padding.

The **margin** box is the last one and defines the space between this box and other elements. (Space outside the box).

In the standard box model, the height and width attributes define the size of the content box. Any padding and border is then added to that width and height to get the total size of the box.

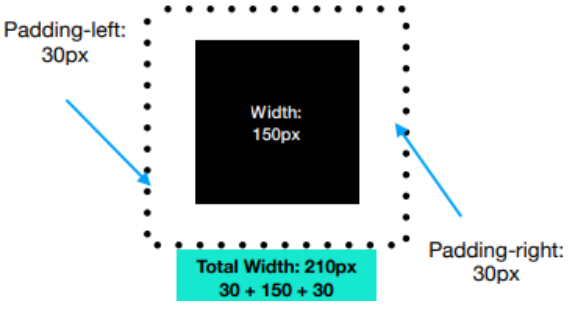
For example, the total width of the box with paddings will be the sum of width plus padding left and padding right. Margin is not counted in this model due to it not affecting the box space itself, rather the space outside of the box.

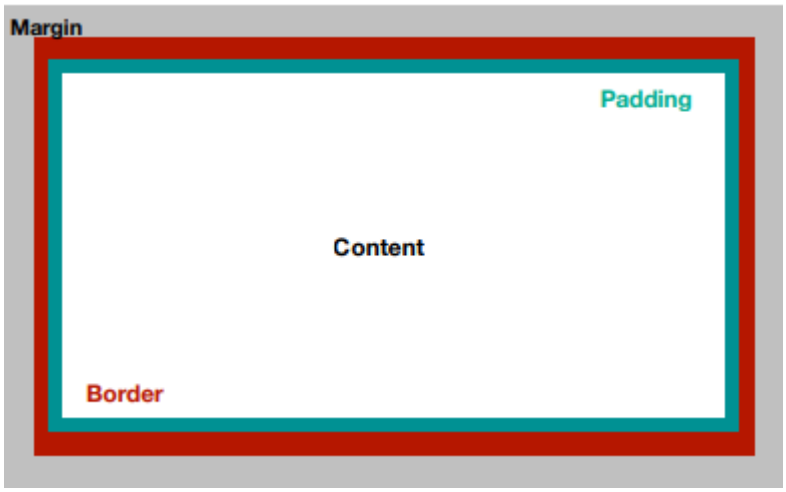
Margin and **padding** can both be styled in the format **property: left, top, right, bottom;**. Example: margin: 5px 10px 5px 10px; . This changes the top and bottom margins to 5px and the left and right margins to 10px.

Elements can get border styling, using the border property, this allows you to display the container outline for elements. The border is always there BUT it is not visible until the border is defined.

Borders can have 8 different styles: dotted, dashed, solid, outset, inset, ridge and groove. Example: border: thickness, style, color; border: 4px dotted blue; (apply this to an element from your project to test it out).

Border styling is similar to margin and padding styling as far as styling top, bottom, left, right. Each side of an border an have its own style. By setting the border radius, you can round the corners of elements with pixels or percentage. You can style each corner separately or using one value for all corners. Radius applies even if the border is not visible.





BACKGROUND

The background-color property sets the background color for an element. In your CSS document, if you have one open still, lets style the body element. Simply type in CSS body { background-color: yellow; }. Background colors can be set using hex, rgb or color names.

Using the background-image property allows you to set the background of an container or page to an specific image. The image address is placed in quotes using the url() attribute.

Example: **body { background-image; url('images/test.png'); }**

Naturally your background image will be tiled, the following rules can be applied in the styling to work with that**:no-repeat, repeat-x**, **repeat-y**, **repeat**(x=horizontal, y=vertical).

Applying the **background-position** property sets the position in which the background image appears. The position property uses coordinate systems to place the image, its start point is always (0,0). **Background-position** can take the values: pixel/percentage, units and keywords. The **background-size** allows the styling for how to fit the image in the box. 2 values are used for this **cover** and **contain**. Cover, covers the entire box while keeping the images aspect ration. Meanwhile contain, fits the image inside the container completely.

LIST STYLING

By default list items are styled with bullet points and ordered list are marked with numbers. Using **list-style-type** you can change the list markings from circles, squares images or nothing at all. Using **list-style-image** you can have an image as an list marker.

Example: **list-style-type:** url **("image.png");** . You can also style where the position of markings should go whether that's inside or outside. Using list-style property is a short for list-style: style-type style-position style-image.

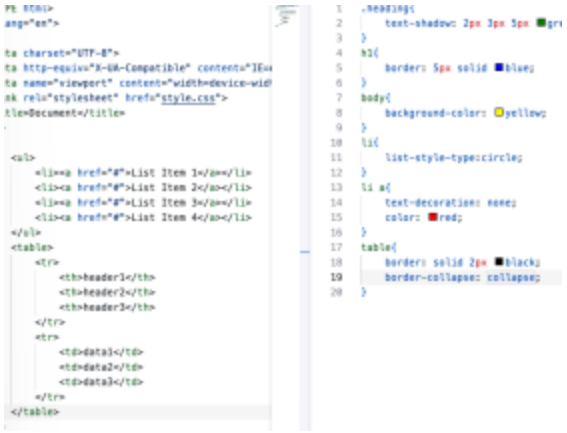
LINK STYLING

Default styling for links are blue in color and underlined. In CSS you can change these properties in many ways using a. Just the letter ' a'. Link styling can be specific to the "state" of the link, a:link - defines the style for normal unvisited links. a:visited - defines the style for visited links. a:active - a link becomes active once you click on it. a:hover - a link is hovered when the mouse is over it. These are known as **pseudo selectors**. Using **text-decoration** we can change the styling of an link or remove the underline, by using **text-decoration:none;** . You can add background color to images when you hover with the a:hover { background-color: value } **selector** and **property**. Common practice is to use links as buttons, so normally you would place your links in side of your list items and style each link using li a **{property:value};**

TABLE STYLING

When styling tables in HTML/CSS, due to the unattractive presentation of tables without styling, it's good to make sure your table is as sleek in design as possible. We can use the **border-collapse** property to make all touching borders collapse into 1 border. This is going to sharpen the look of your border of the table. Just like everything else in HTML the cells, td & th, can receive padding styling for more comfortability in the table. Background color is also available for the cells or table styling. Another pseudo selector that comes in handy with rows and columns in your table, is the **nth-child()** selector. This selector allows you to chose specific rows by **odd** or **even**, and if you wanted by specific **numbers**!

Example: **th:nth-child(even)** { property:value } (Selects all even columns in the table.)



FORM STYLING

Forms can be styled by styling the: form, input, or label. Add some styling to the input and label. Noice that the button will also receive the same styling as it is an input element as well.

Example:

```
input { border: 2px solid red; width:100%; padding: 4px; }
label { font-weight: bold; color:blue; }
```

We can target specific elements inside of our inputs. For example if I want to only target the “name” element, I would use the following code: input[id=‘name’] { property:value; }

These elements can also be styled just as we would a paragraph or heading etc. Applying text to our button is the same exact way, it is definitely best practice to style your button in a way that is attractive and stands out. Pseudo selectors work with these elements as well.

```
<form>

    <label for="name">Name: </label>
    <input type="text" id="name" placeholder="Jon Doe">

    <label for="email">Email: </label>
    <input type="email" id="email" placeholder="name@domain.com">

    <input type="button" value="send">

</form>
```

Form example in HTML.

LAYOUT STYLING

In HTML we have, **block** and **inline** elements.

Block elements extend 100% width of their parent, starting at a new line. **Inline** elements are placed inside block elements, similar to their content and sit on the same line. The **display** property changes if elements value are block or inline. Another available value for **display** is the **flex** value. Flexing in the display property allows us to arrange our elements into rows and columns. By default flex items have equal spacing, you can give items the flex value of 1 or 2 to distinguish spacing amongst all of your items. We will use the flex layout for creating our website at the end of this book so for now, just understand what is being said prior to practice.

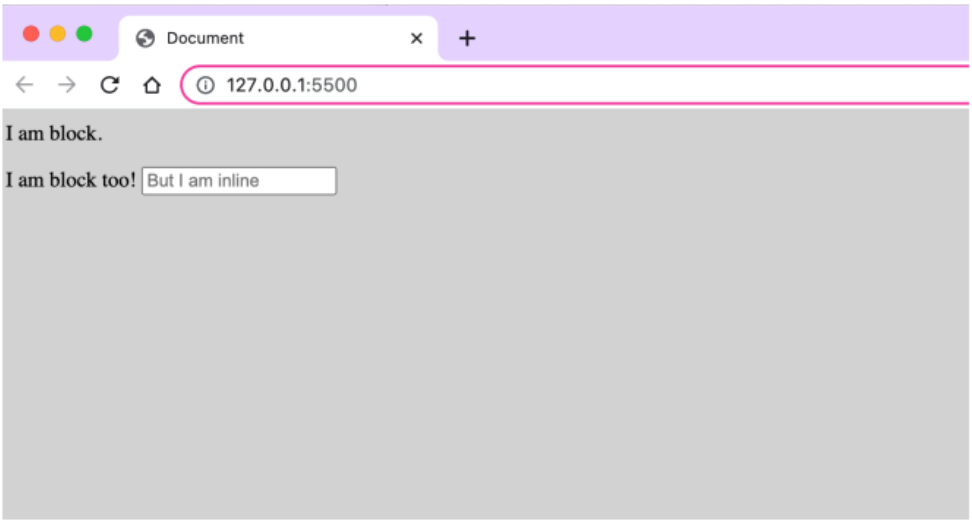
index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
    <header>
        I am block.
    </header>
    <p> I am block too! <input type="text" placeholder="But I am inline"> </p>
</body>
</html>
```

style.css

```css
body{
    background-color: lightgrey;
}
```

Code example for **block** and **inline** elements.



POSITIONING STYLING

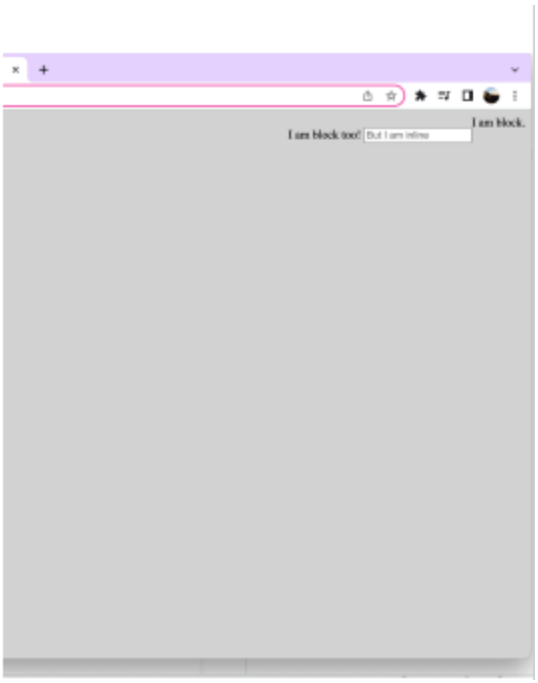
CSS has a really cool property called **float**, this allows you to push an element to the **left** or **right** causing the text to wrap on the opposite side of the floating element. The **position** is also a very useful property in CSS. Default positioning is set to **static**. Properties **top**, **right**, **bottom** & **left** can be used to give direction to the document on how to shift the element. Positions can be:

fixed- causes the element to stick to a specific part of the visible container, which means it is always in the same spot even on scroll

absolute- this removes the element from the normal flow of the page

relative- positioned relatively to its normal position

z-index- allows elements to overlap other elements, this is a stack order of flow or overlay (if positioned outside of the normal flow).



TRANSFORMS STYLING

Transformations in CSS opens the ability to **translate**, **rotate**, **scale** and **skew** our HTML elements. These change position, shape and size. Using the **transform-origin** property allows us to change the position of elements as well. This is simply done by: **transform-origin: 20% 50%;** (You can place any value you want in here).

Translate() moves an element from its current position on the **x-axis** and **y-axis**, from its default position in the parent element. **Skew()** method skews an element along the x and y-axis. Skew is another word for **tilt** or **angle**. Essentially we are tilting the element at an angle. **Scale()** increases and decreases element size, with height and width.

Example: **transform: scale(height** (1.3), **width**(1.3)). CSS allows you to use multiple values at once in the transform property styling.

TRANSITIONS STYLING

Transitioning in CSS allows to change values from one to another over a specified duration. The code in the illustration below shows how we can use a div container to practice on an width expanding **transition** when the element is hovered over with our cursor.

The **transition-timing-function** we can use to change the curve of the transition. The default timing value is **ease** if nothing is selected**. ease - the animation starts slowly, then accelerates quickly. ease-in - starts slowly, then accelerates, and stops abruptly. ease-out - starts quickly, but decelerates to a stop**. **ease-in-out - similar to ease, but with more subtle acceleration and deceleration. linear - constant speed throughout the animation; often best for color or opacity changes.**

KEYFRAMES & ANIMATIONS STYLING

We use animations for complex series of movements, and we are allowed to change as many properties and values as we want. We define animations in CSS by using the **keyframes** selector.

@keyframes paddingswitch {

[b]0%[/b] {padding: 2px 5px 2px 5px;}

[b]50%[/b] {padding: 4px 10px 4px 10px;}

[b]70%[/b] {padding: 8px 20px 8px 20px;}

[b]100%[/b] {padding: 16px 40px 16px 20px;}

}

This is an example of an animation, named **paddingswitch**, that changes the padding of an element three times: **1)** when the element is 50% complete **2)** when the element is 70% is complete **3)** and when the animation is 100% complete.

Two key words we can use are **from** and **to**. Instead of the above code we could also do**: @keyframes paddingswitch { 0% {padding: 2px 5px 2px 5px;} 100% {padding: 16px 40px 16px 20px;}. } Or @keyframes paddingswitch { from {padding: 2px 5px 2px 5px;} to {padding: 16px 40px 16px 20px;} }**

To add this animation to an element we would simply use animation-name along with animationduration. So all together the above code will look something like this : div { animation-name: paddingswitch; animationduration: 5s; } @keyframes paddingswitch { 0% {padding: 2px 5px 2px 5px;} 100% {padding: 16px 40px 16px 20px;} }

Just how we did in transition-timing-function we can apply the same properties for **animation-timing-functions**. Animations come with another property **animation-delay**. This defines an delay prior to the start of an animation. Can be done in **seconds (s)** or **milliseconds (ms)**. When using animations **animation-iteration-count** determines the number of times an animation repeats itself. This can be done with a number value or the word **infinite**. Applying **animation-direction** guides the document on how it should apply the animation frame. There are 4 values we can set: **normal - the default value, which means it plays forward from 0 % to 100%**. **reverse - plays the keyframe in an opposite direction from 100 % to 0% alternate - the animation first runs forward, then backward, then forward**. **alternate-reverse - the animation first runs backward, then forward, then backward.** We can also simply use this string of values for our element in CSS:

div {animation: colorchange 3s ease-in 1s infinite reverse;}

Thank You for Choosing Astro Clare Technology!

At Astro Clare Technology, we express our gratitude for choosing to embark on this coding journey with us! Whether you're a dedicated learner or a parent nurturing the next software engineer, your trust means the world to us.

Our mission extends beyond teaching coding skills. We aim to cultivate critical thinking, logical outlooks, problem-solving skills, discipline, and career-ready structures. The evolving landscape of technology should be accessible to all, and we're here to make that happen.

As you progress through the activities, each successful assessment earns you a certificate from Astro Clare Technology, a testament to your accomplishments. Your journey is not just about learning to code; it's about becoming the greatest engineer you can be.

Unveiling CSS

Welcome to the world of Cascading Style Sheets (CSS)! As you delve into the intricacies of styling, remember the power of rules and selectors. Whether you're defining styles for headers, mastering class and ID selectors, or exploring text styling, your journey is filled with creativity.

The CSS journey covers everything from the box model to background styling, list styling, link styling, table styling, form styling, layout styling, positioning styling, transforms, transitions, and animations. Each concept adds a layer to your coding expertise, transforming you into a versatile developer.

Remember, the flex layout and positioning properties give you control, and transformations open a realm of possibilities. Dive into transitions and animations to bring life to your creations.

Your Journey, Your Achievement

Completing this book marks a significant milestone in your coding journey. Your commitment to learning and growing is commendable. As you showcase your best code forward, know that Astro Clare Technology is proud to be a part of your success.

Clarence Scott,

CEO & Founder

Astro Clare Technology