**javaScript:**

* DOM – document object Model
* How does JS run? Every browser has JavaScript Engine
* Html and CSS are static languages that don’t interact with users.
* Js is client side – communicate with web browser not web server
* Js is an object oriented languages
* The “head” tags on html gets loaded before any other parts that’s why the js functions most commonly placed in between the head open and close tags.
* External javascript – save as example.js and then

<script type=”text/javascript” src=“example.js”></script> on the HTML page

* Now ….

Function:

Document.write()

window.alert()

document.getElementById("math").innerHTML = 6+5; 🡪 puts the value in an id

innerHTML 🡪

General:

Comments = // /\* \*/

Variables -> var x =10 for example.

Object -> var car = {type:”honda”, model:”civic”, color:”black”};

<br> 🡪 break tag

Array.join(“separator”)

What is “meta” tag?

JS Events:

* Mouse Click
* Mouse overs
* Mouse out
* Key Fown
* Document load

Questions:

* What is the script tag an html or javascript ?
* What is the <p> in html ?
* What is the type = “text/javascript” and what its goal ?
* What is document.getElementById("math").innerHTML = text ?
* What is <meta charset="utf-8" /> ?
* What is <meta name="viewport" content="width=device-width, initial-scale=1"> ?
* What is the <div> tag?

The <div> HTML element is **the generic container for flow content**. It has no effect on the content or layout until styled in some way using CSS

* What is the <style> in the <head> ?

An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

* Understanding the

            <div class="preview" align="center">

                <img name="preview" src="images/img1.jpg" />

            </div>

* What exactly is .thumbnails img ? what is the img?

HTML & CSS:

Added some extension check again which….

Pexels.com 🡪 dree photos and videos

<https://unsplash.com/> 🡪 web page for downloading images

<https://validator.w3.org> 🡪 check the validity of your web page

<https://jigsaw.w3.org/css-validator> 🡪 CSS validator

<https://necolas.github.io/normalize.css/> 🡪 for different browsers

https://cssgradient.io 🡪 cssgradient

<https://fontsquirrel.com> 🡪 web for fonts

<https://fonts.google.com> 🡪 google free fonts

<http://canihuse.com> 🡪 check which browsers supports …

<https://milligram.io/> 🡪 Milligram.io

<https://getbootstrap.com/docs/5.1/forms/form-control/> 🡪 Bootstrap

Front end:

* HTML- blocks and skeleton 🡪 markup language
* CSS- style of the web page – padding size font bold etc. visual affects 🡪 styling language
* JavaScript- add functionality or behavior 🡪 programming language

DOM – document object model

Developing tools: Ctrl+shift+i

Close the explorer of visual studio : ctrl + b

Prettier: ctrl+alt+f

Prefences-> Format on save

<!DOCTYPE html> 🡪 Tells the browsers that this is an HTML5

Class:

You add class to an element to be more specific when you work with CSS for example.

General:

* !doc + ctrl+space will give a skeleton with the meta element
* <em> 🡪 bold in text element
* Every page should have only one single heading <h1> for example, the <h\_> tells the importance.
* &<entity>.;🡪 for special characters. HTML entities list
* &nbsp; non breaking separate
* Image: object-fit: cover;

http:

* link: <a href="about.html">press it!</a>
* images : <img class="peru" src="images/peru.JPG" alt="a picture from peru">
* videos: <video controls autoplay loop src="videos/ocean.mp4">Your browser doesn't support videos.</video>
* audio: <audio src=""></audio>
* list:

<ul>

<li>About Me</li>

<li>

Courses

<ul>

<li>bla1</li>

<li>bla 2</li>

<li>bla 3</li>

</ul>

</li>

<li>subscribe</li>

</ul>

* short cut: ol>li\*3
* Tables: thead, tbody, tr, th, colspan, tfoot,
* Search engines thead tbody
* Generic container: Div
* Generic container: Span: with inline element
* Semantic container: article -> figure figcaption mark time
* Wrap 🡺 ctrl+shift+p -> wrap->

Mosh Summary:

•The <head> section is used to provide information about a webpage.

•The <p> element is used to represent a paragraph. A paragraph can be one or many lines of text.

•The <em> element is used to define emphasized text. By default, emphasized text is displayed in italic.

•The <strong> element is used to represent important content. Browsers, by default, render strong content in bold.

•The <i> and <b> elements are considered deprecated because HTML should not be used for styling. That’s the role of CSS.

•Headings are represented using <h1>, <h2>, <h3>, <h4>, <h5>, <h6>. Every web page should have one and only one <h1> element. Headings should have a natural hierarchy and should not be skipped.

•Entities are used to display special characters such as angle brackets, copyright symbol, etc. The most important entities are: &nbsp; (non-breaking space), &lt; (less than sign), &gt; (greater than sign) and &copy; (copyright symbol).

•The <a> (anchor) element, with its href attribute, is used to create a hyperlink to web pages, locations in the same page, files and email addresses.

•A relative URL specifies the target resource relative to the current resource. An absolute URL specifies the location of a resource irrespective of the current resource. It can start with a / to indicate the root of the website or a protocol (eg http://) to represent a resource on a different website.

•The <img> element is used to display an image. It’s a common best practice to set the alt (alternative text) attribute. This helps visually impaired people understand the page content. Also, if the image cannot be loaded, the alternative text is displayed.

•The <video> and <audio> elements are used to display video and audio. These elements have boolean attributes such as controls, autoplay and loop.

•The <ul> element is used to represent a list where the order of items doesn’t matter. The <ol> element is used to represent an ordered list of items. The <dl> (description list) element is used to implement a glossary or to display metadata.

•The <table> element should only be used to represent tabular data. A table can have zero or more <tr> (table row) elements. Each <tr> element can have zero or more cells. Cells can be data cells (<td>) or header cells (<th>).

•The <div> and <span> elements are generic containers used for styling purposes. Divs are block-level elements, spans are inline elements. A block-level element starts on a new line and takes up the entire available horizontal space.

•Semantic elements help us write markup that is more meaningful and descriptive to search engines, screen readers and other software. So, use <div> and <span> elements when no other semantic element is appropriate.

•The semantic elements in HTML5 are: <header>, <footer>, <nav>, <main>, <aside>, <article>, <section>, <figure>, <time> and <mark>.

**CSS:**

* To link CSS with HTML use <link rel="stylesheet" href="styles.css">
* Three ways:

1. Embedded
2. External
3. Inline <p style="color: blue; font-size: 10px;">Lorem ipsum dolor sit amet.</p>

* <p id="exampleP">just checking ....</p> 🡪 on the CSS file #exampleP{ … }
* Normalized css
* What is the different between ID and Class ?

We cant have multiple elements with the same id but we can do it with the same class

* Selectors
* Pseudo class selectors
* Last of type, last child …
* A:visited
* Pseudo elements selectors p::first-letter{} in the css file
* תמונה שמכילה טקסט

  התיאור נוצר באופן אוטומטיArticle>h1.highlight 🡺 (shortcut)
* Selector: First: ID , Second: class , Third: element selector
* !importatnt 🡪 ignore more specific rules
* Color picker on google to get the color number
* Color 🡺 RGB, hex, hsl …
* Linear gradient 🡺 combine colors
* Gradient generator website that helps with css code for gradient 🡺 cssgradient.io
* Borders: thickness, style, color /\*trbl 🡪 top right bottom left\*/
* Css shapes tricks
* Shadows 🡺 right down soft color

CSS mosh summary:

Embedded stylesheets

External stylesheets

Hexadecimal colors

HSL colorsInheritance

Inline styles

Linear gradients

Normalizing CSS

Pseudo-class selectors

Pseudo-element selectors

Radial gradients

Relational selectors

RGB colors

Selectors

Selectors specificity

Separation of concerns

CSS styles can be embedded in an HTML document, written in a separate file (as an external stylesheet) or written inline in an HTML element using the style attribute.

•Inline styles overwrite embedded styles which in turn overwrite external styles.

•External stylesheets provide the best separation of HTML and CSS code and result in more maintainable code. Plus, an external stylesheet can be used in many HTML documents.

•We can select elements by their type, class, attribute or ID.

•Relational selectors help us select elements without the need to assign them a specific ID or class. This, however, can result in fragile styles. If we move elements around, our CSS rules may break. We can still use them in situations where we are certain about the location of elements.

•We can take advantage of pseudo-classes to target elements without the need to give them a specific class. The most common pseudo-classes are: first-child, first-of-type, last-child, last-of-type and nth-child. Pseudo-classes start with a single colon.

•With pseudo-elements we can style a part of an element. The most common pseudo-elements are: first-letter, first-line, selection, before and after. Pseudo-elements start with double colons.

•Selectors specificity determines the weight of a selector. When multiple selectors target the same element, the browser applies the selector with the higher specificity (weight). If two selectors have the same specificity, the one that comes last is the winner.

•ID selectors are the most specific selectors because we cannot have multiple elements with the same ID. Class and attribute selectors are less specific because we can have many elements with the same class and/or attributes. Element selectors are the least specific selectors.

•In VSCode, we can see the specificity of a rule by hovering our mouse over it. The specificity is represented using three numbers (x, y, z) where x represents the number of ID selectors, y represents the number of class/attribute selectors and z represents the number of element selectors.

•Some CSS properties inherit their value from their parent element. Typically, properties that are used for styling text such as text color, font, font size, etc are inherited. We can stop the inheritance by setting the value of a property to initial. To enforce inheritance, we should set the value of a property to inherit.

• We can specify colors by their name, hexadecimal value, RGB/RGBA value or HSL/HSLA value.

* RGBA and HSLA values include an alpha channel used for transparency. The value for the alpha channel is a decimal point number between 0 (completely transparent) and 1 (completely opaque).

•Using the linear-gradient() and radial-gradient() functions we can create gradients in CSS. Gradients are images so they cannot be used as the value of background-color property. We can use them as the value of background-imageor background properties.

•The border property is a shorthand property for border-top, border-right, border-bottom and border-left. It takes three values: the thickness of the border, its style and its color.

•We also have specific properties like border-width, border-style and border-color. These properties take four values for the top, right, bottom and left borders.

•Using the box-shadow and text-shadow properties we can apply a shadow to elements and text. These properties take a few values. The first two values determine the horizontal and vertical distance of the shadow from the element. The third value (called blur radius) determines the softness of the border. We can specify the color as the fourth value.

* CSS continue -
* Box mode: content🡪 padding 🡪 border 🡪 margin
* Sizing element: box-sizing
* \*, \*::before, \*::after …..
* Span 🡪 display inline-block
* Overflow: visible,hidden,scroll,auto…
* Measurement : px , %, vw, vh ,em, rem …
* תמונה שמכילה טקסט

  התיאור נוצר באופן אוטומטיPosition
* Float
* Floated elements are invisible to the parent element
* Clear
* display
* What is flex-basis?
* Flex box
* תמונה שמכילה טקסט

  התיאור נוצר באופן אוטומטי
* Grid – repeat(number , px)
* Gap
* Grid area

**Mosh summary:**

* When rendering an HTML document, the browser puts each element inside a box. The box contains four areas: the content area, the padding area, the border area and the margin area.
* Padding is the space between the border and the content area. Margin is the space outside of an element and should be used to separate elements from each other.
* Margin collapsing happens when the top and bottom margins of elements are combined into a single margin. The size of the margin is equal to the largest of the two margins.

There are two types of HTML elements: block-level and inline.

* Block-level elements always start on a new line and take up the entire available horizontal space. The <p> and <div> elements are examples of block-level elements.
* Inline elements don’t start on a new line. They take up as much width as necessary. The <span>, <a> and <img> are a few examples of inline elements.
* We can size elements by setting their width and height properties. These properties have no effect on inline elements. To size an inline element, we need to set its displayproperty to inline-block.
* By default, the width and height properties are applied to the content box. So paddings and borders increase the size of the visible box. This behavior can be changed by setting the box-sizing property to border-box.
* Overflow occurs when an element’s content is too large to fit. Using the overflowproperty we can specify what should happen when overflow occurs.
* Measurement units in CSS fall into two categories: absolute and relative units. Examples of absolute units are px, pt, in, cm, etc. Examples of relative units are %, vw, vh, em and rem.
* Using the position property we can precisely position an element. The default value of this property is static. If we change the value of this property, the element is considered positioned.
* By setting the position to relative, we can position an element relative to its normal position. By setting it to absolute, we can position it relative to its positioned parent. That means, the parent (or ancestor) should be a positioned element. By setting the position to fixed, we can position the element relative to the viewport.
* By setting the float property, we can push an element to the left or right side of its container. Other elements will flow around the floated element and fill the available space.

Floated elements are invisible to their parent. This behavior is called collapsing parent and often causes layout issues. To fix this, we have to clear the floated elements.

* The Flexible Box Layout (or FlexBox or just Flex) is used for laying out elements in one direction (in a row or column). A common application of Flex is in building navigation menus.
* The Grid Layout is a two-dimensional grid system. It’s often used to lay out major page areas, photo galleries, etc.
* With media queries we can provide different styles for different devices depending on their features such as screen size, orientation, etc. The most common application of media queries is in providing different styles based on the viewport width.
* By using media queries and relative measurement units we can build responsive web sites that adjust smoothly to various screen sizes.

**CSS FONT Styling:**

* @font-face
* At fontsquireel you download the font you like, it will download it as ttf file then you upload it to download it as woff and woff2 files because its lighter file. After that you open folder and upload the woofs files and copy the @font-face from the stylesheet file to your css file and make few changes
* Font-display: optional …
* Google fonts
* Fonts-size 🡪 best is rem
* Set the font-size to a relative html element ( 62.5%)
* Type-scale.com 🡪 web for font scale
* Law of proximity
* Line-height 🡪 recommended 1.5 of the font-size
* Line-length between 50-70 characters
* Width: 50ch

**Mosh Summary:**

•Typography is the art of creating beautiful and easy-to-read text. Given that 95% of the content on the web is text, as a front-end developer, you must ensure that the text on your web pages is easy to read and visually appealing on various screen sizes.

•Fonts fall into three main categories: Serif, Sans-serif and Monospace. Serif fonts have a line/stroke at the edges of their characters. They’re more professional and serious. Sans-serif fonts don’t have those edges. They’re more modern, warm and friendly. Monospace fonts have equal-width characters. They’re often used in displaying code.

•The default color for the body text is pure black (#000). It’s best to change it to dark grey.

•We can use the font-family property to set the font for an element. We should set this property to a font stack which contains multiple fonts as fallbacks.

•In the past, we used web safe fonts because they’re available on almost all computers. These days, however, we can easily embed custom fonts.

•Font files come in a variety of different formats: TTF, OTF, EOT, WOFF and WOFF 2.0. Out of these, WOFF and WOFF 2.0 are recommended for the web because they’re more compressed and can be downloaded in less time.

•We can convert any font file to a WOFF file on fontsquirrel.com.

•To embed a custom font, we should first register it using the @font-face rule.

•When using a custom font, the user may experience a flash of unstyled text (FOUT). Some browsers display text using a fallback font while downloading the custom font and swap it once the custom font is available. This may cause a layout shift depending on how the content is structured. Some browsers hide the text initially while downloading the custom font. This causes a flash of invisible text (FOIT). Using the font-display property we can tell the browser how to handle this situation.

•Using font services we can get access to thousands of beautiful fonts with zero or minimal cost. Google Web Fonts is the most popular and free font service. When using these services, fonts and @font-face rules are served from the provider’s servers.

•A common practice for content-heavy websites is to use the system font stack which represents the default font used by an operating system. With the system font stack, we achieve a better performance because no fonts need to be downloaded and the FOUT/FOIT doesn’t happen either. Plus, the page looks more familiar to the user because they see the same default font used by their device. On the flip side, the default font varies from one device to another.

•It’s best to size fonts using the rem unit. This will set the font size relative to the font size of the root (html) element. Using media queries, we can resize the base font size, and as a result, the font size for all elements will be re-calculated with no extra code.

•It’s best to use the rem unit for vertical margins. For headings, the top margin should be noticeably greater than the bottom margin so the heading gets separated from the text before and gets connected to the text after.

•The law of proximity describes how humans perceive the connection between objects. Objects that are closer are perceived to be related.

•Using the line-height property we can specify the height of lines. It’s best to set this property to a unitless value around 1.5. This value will be multiplied by the font size of the current element so we don’t need to remember to change the line height if we modify the font size.

•The three properties used for horizontal spacing are: letter-spacing, word-spacing, and width. It’s often better to apply a negative letter spacing to headings so they look more compact.

•The ideal line length is about 60-70 characters. We can achieve that by applying a width of 50ch. The ch unit represents the width of the 0. 50 zeroes roughly represents 60-70 characters because some characters (like i and 1) are more narrow than 0.

•Using the Network tab in Chrome DevTools, we can simulate a slow connection. This is called Network Throttling.

•Browsers store some assets behind web pages in a permanent storage called cache. It’s essentially somewhere on the disk. The cache can always be cleared.

**CSS Cheat Sheet:**

**Styling Fonts**

font-family: Arial, Helvetica, sans-serif;

font-size: 1rem;

font-weight: bold;

font-style: italic;

**Vertical Spacing**

margin: 3rem 0 1rem;

line-height: 1.5;

**Horizontal Spacing**

letter-spacing: -1px;

word-spacing: 2px;

width: 50ch;

**Formatting Text**

text-align: center;

text-indent: 1rem;

text-decoration: underline;

text-transform: uppercase;

white-space: nowrap;

direction: rtl;

**Multi-column Text**

column-count: 2;

column-gap: 2rem;

column-rule: 3px dotted #999;

**CSS Images:**

[**https://www.responsivebreakpoints.com/**](https://www.responsivebreakpoints.com/) **🡪 resizing images**

[**https://fontawesome.com/kits/8cbc7a5659/use?welcome=yes**](https://fontawesome.com/kits/8cbc7a5659/use?welcome=yes) **🡪 icons**

* Raster and vector
* Transparency
* Cssspritetool 🡪 make many images to only one to get solid http request
* Css sprite
* Data uri generator 🡪 binary data of an image
* Css clip generator 🡪 cut images as you want
* Css filter function 🡪 add filters to an images
* Srcset 🡪 options in an img
* Responsive image breakpoints

Mosh Summary:

•Images fall into two categories: raster and vector. Raster images are made up of pixels. Vector graphics are defined by a set of mathematical vectors (eg lines and curves).

•Raster images often come from cameras and scanners but they can also be produced in software. Any file with the extension of JPG, PNG, GIF is a raster image. We use these images for displaying photos. •Vector images are exported from drawing tools like Adobe Illustrator. Files with the SVG extension represent vector graphics.

•We use the img element to display content images. Content images can represent meaningful content or be used for decorative purposes. If used for decoration, we should set the alt attribute to an empty string; otherwise, screen readers will read out the name of the file which may be distracting to the user.

•Using CSS sprites we can combine multiple images into a single image (sprite) and reduce the number of HTTP requests. The problem with CSS sprites is that every time we need to change one of the images in the sprite, we have to re-generate the sprite. So, use this technique for small images that don’t change often. You can generate a sprite using <https://cssspritestool.com>.

•Data URLs allow us to embed an image data directly in an HTML document or a stylesheet. The embedded code is always greater than the size of the original resource and makes the document convoluted and hard to maintain. Use this technique only if you know what you’re doing!

•We can clip an image using the clip-path property in CSS. To generate a clip path from basic templates, visithttps://bennettfeely.com/clippy

•Using the filter property in CSS, we can apply filters such as grayscale, blur, saturate, brightness and so on.

•High-density screens like Apple’s Retina displays contain more pixels than standard-density screens. The pixels on these screens are smaller than the pixels on standard-density screens. So when displaying an image, the screen uses a scale factor (1.5 or greater) to scale up the image. As a result, raster images may look a bit blurry when shown on these screens. To solve this problem, we can supply 2x or 3x versions of an image using the srcset attribute of the img element.

•For flexible-sized images, we need to supply the image in various sizes for different devices like mobiles, tablets and desktop computers. If we supply a single image, the browser on each device has to resize the image which can be a costly operation. The larger the image, the more memory is needed and the more costly the resizing operation will be. Plus, the extra bytes used to download the image will be wasted. This is the resolution switching problem. To address this, we should give the img element a few image sources and the size of the image for various viewports. The browser will take the screen resolution and pixel density into account and download the image that best fits the final size.

•We can use https://responsivebreakpoints.com to generate our image assets for various screen sizes.

•WebP is a modern image format created by Google and is widely supported except in Internet Explorer. To support modern image formats, we can use the picture element with multiple sources. The picture element should always contain an img element otherwise the image is not shown.

•Sometimes we need to show a zoomed in or a cropped version of an image for certain viewport sizes. This is the art direction problem. To handle this, we use the pictureelement with multiple sources. Each source should contain a media condition and a srcset. The browser will pick the first source whose media condition matches.

•Scalable Vector Graphics (SVG) files are great for logos, icons, simple graphics and backgrounds with patterns. They are often very small and can scale without losing quality. You can get find plenty of beautiful SVG backgrounds on <https://svgbackgrounds.com>

•We can also use icon fonts for displaying icons. The most popular icon fonts are Font Awesome, Ionicons and Material Design Icons.

**CSS Cheat Sheat:**

**Background images:**

background: url(../images/bg.jpg);

background-repeat: no-repeat;background-position: 100px 100px;

background-size: cover;

background-attachment: fixed;

**Clipping**:

clip-path: polygon(50% 0%, ...);

**Filters**:

filter: grayscale(70%);filter: blur(3px);

filter: brightness(0.5);

filter: contrast(200%);

filter: drop-shadow(10px 10px 10px grey);

filter: hue-rotate(90deg);

filter: invert(50%);

filter: saturate(25%);

filter: sepia(50%);

filter: opacity(50%);

filter: grayscale(70%) blur(3px);

**supporting High-density Screens:**

<img srcset=“

images/meal.jpg 1x,

images/meal@2x.jpg 2x,

images/meal@3x.jpg 3x ”

src=“images/meal.jpg”/>

**Resolution Switching:**

<img srcset=“

images/meal.jpg 400w,

images/meal@2x.jpg 800w,

images/meal@3x.jpg 1200w

“sizes=“

(max-width: 500px) 100vw,

(max-width: 700px) 50vw,

33vw “

src=“images/meal.jpg”/>

**Supporting Modern Image Formats**

<picture>

<source type=“image/webp” srcset=“...” />

<source type=“image/jpeg” srcset=“...” />

<img src=“...” alt=“...” />

</picture>

**Art Direction**

<picture>

<source media=“(max-width: 500px)”

srcset=“...” />

<source media=“(min-width: 501px)”

srcset=“...” />

<img src=“...” alt=“...” />

</picture>

**Form CSS:**

* Form 🡪 input 🡪 label
* The “for” with the “id” connect the box with the tag
* Button type=”reset”
* input[type="text"] 🡪 only input with type = text
* transition 🡪 delay the time of appearing
* framework 🡪 Bootstrap, foundation, semantic UI, ….
* <https://getbootstrap.com/docs/5.1/forms/form-control/> 🡪 On boostrap docs we can take code and classes names into out project
* <https://milligram.io/> 🡪 Milligram.io
* HTML input element 🡪 w3school or mozzila
* Textarea 🡪 creates a text area
* Placeholder🡪 put an “Email” until you enter an email
* Datalist option 🡪 <input type="text" list="countries" />
* Dropdowm list 🡪 select -> option
* Dropdown list (optgroup) label
* Radio buttons
* Validation
* Formspree.io/forms 🡪 backend server
* Label 🡪 create the “label” outside the field
* What is the “name” in input means?

**Mosh Summary:**

•Label elements make our forms more accessible. When the user clicks on a label, the associated input field gets focus.

•We use the input element to accept data from the user. The type attribute of the input element determines the type of the input field and can be text, password, email, checkbox, radio, range, date, file, etc.

•Using data lists we can provide a suggestion list (autocompletion).

•To display a drop-down list, we use the select element. This element can contain one or more option elements.

•The fieldset element (along with a legend) is used to group related input fields. Alternatively, we can use the section element to group related fields.

•Hidden fields are not visible to the user and are often used to specify the ID of the content being edited. Don’t use them to store sensitive information because they can easily be accessed via the source code of the page

•Always validate your form data to prevent security attacks and data corruption.

•HTML5 comes with built-in validation. The most common validation attributes are required, minlength, maxlength, min, andmax. •To submit a form, you should set the action and the method of the form. The action attribute repress

ents where data is sent. The method specifies how the data is sent and can be either GET or POST.

•With the POST method, form data is included in the body of the request. With the GET method, form data is included in the URL as query string parameters. That’s why the GET method is often used when we need to enable bookmarking pages. In contrast, the POST method is used when we need to update the data.

•Use https://formspree.io to create a backend for your forms

**Animation Css:**

* <class>:hover
* Transform
* Prespective :200px ?
* Keyframe 0% 25% … each frame has it own style
* The key frame make it more automatic way
* [https://animate.style/](https://animate.style/%20) 🡪 animation website

**Mosh Css sheat:**

**Transformations**

transform: rotate(15deg);

transform: rotate(-15deg);

transform: scale(1.3);

transform: skew(15deg);

transform: translate(10px, 20px);

transform: translateX(10px);

transform: translateY(20px);

transform: rotate(15deg) scale(1.3);

**transitions:**

transition :transform .5s;

transition: transform .5s ease-out;

transition: transform .5s ease-out 1s;/\* 1s delay \*/

transition: transform .5s, color .3s;

**Animations:**

@keyframes pop {

0% { transform: scale(1); }

50% { transform: scale(1.5); }

75% { transform: rotate(45deg); background: tomato; }

100% { transform: rotate(0); }}

.box {

animation: pop 3s ease-out;

}

**Mosh Summary:**

•Using the transform property, we can apply one or more transformations to an element.•The most common transformation functions are rotate(), skew(), translate()and scale(). •The transition property is used to animate one or more properties.•To create a custom animation, first, we need to define the keyframes. Each keyframe includes the list of styles to be applied at a given moment in time. Once we define the keyframes, we can use the animation property to animate an element.

**Mosh Summary:**

•Follow a naming convention for naming IDs and classes. The most common naming conventions are PascalCase, camelCase and kabob-case.

•For a small project, you can write all of your CSS rules in one stylesheet. Use CSS comments to create logical sections in your stylesheet. For a more complex project, you need to separate your stylesheet into multiple files and combine them together using build tools like Webpack, Rollup or Parcel.

•Avoid over-specific selectors. Limit nesting to two or maximum three selectors.

•Avoid the !important keyword as much as possible.

•Sort CSS properties. This makes it easier to read your code.

•Take advantage of style inheritance and reduce duplication in your styles.

•Use CSS variables, also called custom properties, to keep your code DRY.

•We often declare variables using the :root selector that targets the html element. We can then access these variables using the var() function.

•Object-oriented CSS is a set of principles for creating reusable components. The two principles in object-oriented CSS are: 1- Separate container and content. 2- Separate structure and skin.

•BEM (Block Element Modifier) is a popular naming convention for CSS classes.

Extensions:

* Css peek
* Html and css support
* Prettier
* highlight matching tag
* todo highlight

**Repeat:**

* normaloziztion
* raster images and the differences
* transition
* position: relative
* transform: perspective(200px) translateZ(-50px); 🡪 first need to give perspective from the object to us **/\*for translate \*/**