**HTML**

* <hr> tag is the alternative way for <br> tag to create a line break in HTML
* HTML ignores spaces and this process is called whitespace collapsing spaces created by the space bar, the tab key and the return key are all ignored when we write our code
* &nbsp; is used for spacing in HTML.
* <em> tag is used for Italic text. <strong> tag is used for bold text. <strike> tag is used for strike text. <u> tag is used for underlined text.
* <li> tag is used for bullet points in an unordered list, by using CSS we can change the bullet style in an unordered list. In ordered list <li> tag is used for numbered list.
* <video> tag is used to embed video in the HTML document. The ***controls*** attribute is used to specify if we want the user to be able to see the control menu. If we remove the controls attribute we have to add the autoplay attribute otherwise the video will not play automatically, user will have no way to play the video or save it. The auto play attribute does not work on mobile devices like iPods and iPhones.
* <a> tag is used to link any links to an image, text etc. We need to add a target attribute to our <a> tag to open the page in a new browser window. <a> tag also links to a specific content area on our existing page.
* The <span> tag is an inline container used to mark up a part of a text, or a part of a document. The <span> tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute. The <span> tag is much like the <div> element, but <div> is a block-level element and <span> is an inline element.
* ***Absolute File Referencing*** – Includes the full path/URL of the file
* ***Relative File Referencing*** - Includes the path of the file is defined relative to its directory
* If we don’t specify the percentage in the width or height, that converts into pixel. Pixel is fixed, won't adjust based on browser size.
* ***iframe*** is an HTML document embedded inside another.
* The HTTP protocol facilitates communication between our web browser and server.
* The Get requests can be cached, remain in browser history, should only be used to retrieve data, have length restrictions, should not be used for sensitive data, can be bookmarked.
* The POST requests have no restrictions on data length, cannot be bookmarked, do not remain in browser history, cannot be cached.
* DOM is best defined as a tree of objects created by your web browser.
* When the Web page loads your web browser uses the DOM to interpret that page and display its contents

**CSS**

* There are three types of CSI styles internal, inline, and external styles.
* ***Inline CSS*** is basically just writing the CSS code inside essentially of the element.
* ***Internal style*** sheet simply means that the style sheet is contained in the same HTML document in which the styles are used.
* ***External style*** sheet is placing our style rules in a CSS file and linking to it.
* ***Inline style*** is the style rules that are added directly inside the element.
* ***Class selector*** can be used more than once in a document whereas ***Id selector*** can be used once in a document.
* ***Padding*** create spacing between the perimeter of each team out element in the content contained within it.
* ***Relative Positioning*** allows specification of the top, right, bottom, and left position of an element. With relative positioning, the div moves with the flow of the page.
* ***Absolute positioning*** will ensure the element stays in the exact same position specified in the style rule. It does not consider the flow of the page when you resize the browser window.
* ***Fixed positioning*** fixed position is similar to absolute positioning. The difference is that with fixed positioning the element does not scroll with the page.
* The ***float property*** is used to wrap images or any element around HTML elements.
* When we mix multiple elements inside of a parent element, then ***pseudo-classes*** don't work really well. They are however perfect for situations like where all the child elements are the same.
* Priority of style Declaration marked !important 🡪Inline Styles 🡪 ID selectors 🡪 Class or pseudo class(:) 🡪Element Selector 🡪Universal selector(\*)
* ***Collapsing margins*** - when we have two margins that occupy the same space, only one of them is actually visible on the page. And that is usually the larger of the two.
* ***Block Level Elements***
* Elements are formatted visually
* Elements occupy 100% of parent element width, no matter the content
* Elements are stacked vertically by default, one after the other
* **Default Elements** – body, main, header, footer, aside, p, ul, ol, li, div, h1-h6, section, nav etc
* **With CSS** – display:block
* ***Inline Elements***
* Occupies only the space necessary for its content
* Causes no line-breaks after or before the element
* Box model applies in different way: heights and widths do not apply
* Padding and margin are applied only horizontally(left and right)
* **Default Elements:** strong, em, a, button etc
* **With CSS:** display:inline
* ***Inline Block Elements***
* Looks like inline from the outside, behaves like block level on the inside
* Occupies only contents space
* Causes no line breaks
* **Default Elements:**img
* **With CSS:** display: inline-block
* ***Normal Flow***
* Default Positioning
* Element is in flow
* Elements are simply laid out according to their order in the HTML code
* **Default Positioning:** position:relative
* ***Absolute Positioning***
* Element is removed from the normal flow: out of flow
* No impact on surrounding elements, might overlap them
* We can use top, bottom, left or right to offset the element from its relatively positioned container
* **With CSS:** position:relative
* ***Floats***
* Element is removed from the normal flow: out of flow
* Text and inline elements will wrap around the floated element
* The container will not adjust its height of the element
* **With CSS:** float: left , float:right
* ***Flexbox***
* Flexbox is a set of related CSS properties for building 1-dimensional layouts
* The main idea behind it was to basically allow browsers to automatically divide empty space inside some container element, by its child elements.
* Flexbox makes it easy to automatically align items to one another inside a parent container, both horizontally and vertically
* Flexbox solves common problems such as vertical centering and creating equal-height columns
* Flexbox is perfect for replacing floats, allowing us to write fewer and cleaner HTML and CSS code
* ***CSS Grid***
* CSS Grid is a set of CSS properties for building 2-dimensional layouts
* The main idea behind CSS Grid is that we divide a container element into rows and columns that can be filled with its child elements
* In two dimensional contexts, CSS Grid allows us to write less nested HTML and easier to read CSS
* CSS grid is not meant to replace flexbox! Instead, they work perfectly together.Need a 1D layout? Use Flexbox. Need a 2D layout? Use CSS Grid.

**JavaScript**

* Javascript variables are case sensitive(x & X are different variables).
* Reserved words like Javascript keywords cannot be used as variable names.
* Strings must be written in single or double quotes numbers can be written without quotes.
* The order of computations follows the following sequence brackets, exponents, division, multiplication, addition, and subtraction.