**Web Development**

**HTML & CSS**

Margin – Provides space between border and outer elements

Padding – Provides space between border and the content of the element

***display : block*** – With this display property , an element act as **block** element or paragraph . **It starts on new line and takes the whole width of the container .**

***margin : auto –*** with this property , the element gets **horizontally centered** withinits container

**clear –** this property specifies on which sides of an element , floating elements are not allowed to float

**clear : both -**  no floating elements are allowed on either the left or the right side

**::after** – it is a selector inserts something after the content of each selected elements

* To insert the after/before content , **content** property is used

**display : flex** - we use this to make the flex container flexible

**box-sizing** – This property defines how the width and the height of an element are calculated i.e should they include padding and borders or not .

**box-sizing : border-box** - by this value , the width and height properties include content , padding and border .

- Classes and IDs are **elements selectors .** We can grab/access any element through it

**[class\*=”test”]** – it is an **attribute selector** , matches every element whose class value contain this value(i.e test) and apply its properties to all matched elements .

**\*** - this will select all elements

**Responsiveness** – It is the technique by which we change the layout of the website according to screensize/viewport and resolution .

* **Media queries**
* **Flexbox**
* **CSS grid**

**Block element** – A block element has some white space above and below it.

* It doesn’t tolerate any other element next to it . Ex:- <h> , <p>
* takes up the full width(of its container) available , with a newline above and below it
* to make any element as block we use display property as **display:block**
* Specifically **, Id’s** are used for webpages to interact with Javascript not for styling unlike classes .

**display : relative** - element will be positioned relative to the normal position .

**position : absolute** – the element is positioned absolutely to its first positioned parent

**flex-wrap** – This property decides whether flex items are forced onto one line or can wrap onto multiple lines .

**box-shadow : *horizontal vertical blurr-radius spread-radius color***

**<hr> -**It is used to define a thematic break in an HTML page

**HTML semantic elements** – These elements clearly describe it meaning to both browser and the developer

ex:- <header> , <nav> , <article> , <section> , <footer> etc.

<article> - It is an HTML semantic , which specifies independent and self-contained content . It is used to contain articles types of materials like , Blogs , news-article

**Javascript**

Important keywords/Functions:-

**document** – It is the keyword allows to get access to HTML page .

**getElementById()** – It allows us to grab any element of HTML by ID .

**console.log(token)** – print items

array.**push(element) -** this method is used to insert any array element into the array

array.**indexof(element) –** it gives the index of the given element in an array

array.**splice(index , number of elements)** – it takes off the given index-element from an array

var x = **window.prompt(“Enter = ”) –** it will receive an user-input and will assign it to the variable x

array.**length** – returns the array length .

***OBJECTS***

**objects –** it is the collection of key-value pair

**var student = {firstName: ”Rahul” , lastName:”Kumar” , age : 12};**

**var student = {} ; <- empty object**

**var x = new Object();** - we can create an empty object by this only . Afterwards , we can add key-value pair that empty object

After creating object , we can add key-value to it by following way and same way will be applicable for an empty object.

***x.keyOne = value1;***

***x.keyTwo = value2;***

*- We can create javascript objects manually by functions(constructors) also .*

***function Student(first,last,age)***

***{***

***//down below , we are creating keys/properties explicitly and we are adding their values by function-parameters***

***this.firstName = first;***

***this.lastName = last ;***

***this.Age = age ;***

***this.greeting = function(){***

***return "Hello everyone , this is " + this.firstName +" " + this.lastName + " and I am " + this.Age + " years old . " ;***

***}***

***}***

***var studentOne = new Student(“Rahul”,”Kumar”,85); <- This is the object of Student()***

**-** ***In javascript , we create the objects of a function too like in classes***

variable**.value –** value function modifies or set the value of the variable(inputs)

variable**.innerText** – it works as same as **.value** but for variables holding non-input elements .

- target.**addEventListener(event-type , listener)** – this method allows you to set up a function to be called when a specified event happens , such as when user clicks a button .

- **parseFloat()** method turn a string into float

- **console.log()** and **alert()** methods turns anything into strings automatically

- Default behavior of form is that it **refreshes** itself on every submission and wipe all your data off . To prevent this we off this default feature , using **preventDefault()** method .

event.**preventDefault()** – it cancels the event if it is canceleable , or the default action that belongs to the event will not occur .

**Version Control(GIT and Github)**

**- It is** also called source control .

- Version control is the practice of tracking and managing changes to software code .

- Version control systems are software tools that helps software teams manage changes to source code over time

**Some Terminal commands**

**mkdir** foldername – creates a directory

**touch** filename – creates a file

**vim** filename - open the file into VIM terminal text editor

**:x** - to save and come out of the VIM

**ls** - to check the directory elements

**rm** filename - deletes the file

**GIT commands**

**git status** - checks the status of the working repo

**git init** - initializes new empty repo

**git add .** - it adds the untracked/unstaged changes to the staging area

**git commit -m “message”** - it saves the changes made with a message . It doesn’t save changes in remote servers but only in the local repo of GIT .

**git log** - it gives the history of your previous commits

**git checkout** first7digits - used to go to(or checkout) specific commits

**git checkout** branchName - to switch between branches

**git checkout -b** newBranch - creates new branch

**git branch** - shows the available branches

- **Branch** in GIT is another line of development , which can be modified without affecting the main part of the repo

**git push origin master** - to save/make changes to the remote server like GITHUB