Javascript
- Programming language
- give instructions to computer

Variable and datatypes
- console.log is used to print a message to the console.
console.log("pranjal");

Variables are containers for data
(Dynamically Typed)

Rules-
- Variables names are case case sentitive "a" and "A" is different
- only letters, digits, underscore(_) are $ are allowed (not even space)
- only a letter, underscore (_) or $ should be 1st character
- Reserved words cannot be variables names

camel case (First word First letter capital) Pranjal
(second word second letter capital) Bramhankar
Pranjal_bramhankar (snake case)
Pranjal-Bramhankar (kabab case)
PranjalBramhankar (Pasal case)

let, const, var
Var- Variable can be re-declared and updated
A global scope variables.
let - Variable cannot be re-declared but can be updated
A block scope variables.
const - Variable cannot be re-declared or updated
A block scope variable.

Boilerplate code.

object is a keyvalue.

- Datatypes in Js.
  1. Number
  2. String
  3. Boolean
  4. Undefined
  5. Null
  6. BigInt
  7. Symbol

(All are the primitive data types) - (7)

Non-primitive object (is a collection of values) (arrays, functions)

Js array can hold any type of datatypes.

- Numbers - followers
  String - Name
  Boolean - follow or unfollow
  Undefined - If there is no valus stored in variable they are undefined
  BigInt - let x = BigInt("123");
  = 123n

- For comment in Js (/*) (//)

- Operators -
  - Arithmetic operators.
    +, -, *, /
    modulus (%) (remineder is a modulus) 5%2 = 1
    Exponentiation - power (**) a**b = a^b $5^2 = 25$
  - Unary operator
    - Increment ++ a++ = a+1 (a++) post (++a) pre
    - Decrement -- a-- = a-1
  - array

```
let ioi = [1, 2, "ioistudy", true]
  console.log(ioi);
    letuser 1 = {
                Name: Pranjal
                Age: 19
                }
```

continue - script

- Assign operators

= += -= *= %= **=

a += 1 (a = a+1)

a += 4 (a = a+4)

- comparision operator (True / False)

Equal to ==

Equal to & type ===

Not equal to !=

Not equal to & type !==

>, >=, <, <=

- logical operators < false / True.

logical AnD &&

logical OR ||

logical NOT !

conditi 1	conditi 2	&&	||	!
T	T	T	T	F
T	F	F	T	F
F	T	F	T	F
F	F	F	F	T

- odd even condition

(num % 2 === 0)

- else - If condition

Ternary operators

condition ? true output : false output.

(3 condition If-else)

(2 condition - ternary)

- loops

loops are used to excute a peice of code again and again.

```
for (let i = 1 ; i <= 5 ; i++)
```

initilize — stopping condition — updation

- calulate sum 1 to 5

```
let sum = 0;
For (let i = 1 ; i <= 5 ; i++) {
        sum = sum + i
    } console.log ("sum = ", sum);
```

- Infinite loop : A loop that never ends.

- While loop — stopping condition

```
while (condition) {
        }
```

```
let i = 1;                          (No semi-colun after while)
while (i <= 5) {
    console.log ("i = ", i);
     i++; }
```

- Do while loop (run atleast one time)

```
do {
    //
    } while (condition);
```

```
let i = 20;
do {                                (semi-colun after while)
    console.log ("pranjal");
     i++;
  } while (i <= 10);
```

for of
for in

- For of loop (help in arrays and strings)

```
For (let val of strVar) {
    }
let str = "Pranjal";
 for (let i of str) {          (//iterator -> characters)
      console.log ("i :", i);
    }  size++;
    console.log ("string size =", size);
```

- For in loop (help in objects and arrays)

```
For (let key in obj var {
    }
 let student = {
   name: "Pranjal",
   age : 19,
   cgpa : 9.1,
   isPass : True,
   };
  For (let i in student) {
     console.log(i);
    }
For print key's value
  console.log ("key =", key, "value =", student [key]);
```

Strings
String is a sequence of characters used to represent text.

create string
- let str : "Pranjal"

string length
- str.length

string Indices
- str[0], str[1], str[2]

string create
let str = "Pranjal";

- Template Literals (All are in single strings)
A way to have embedded expressions in strings
'this a template literal'

String interpolation
- To create strings by doing substitution of placeholders
'string text ${expression} string text'

escape character (\n) - next line
(\t) - tap

String method
This is a built-in functions to manipulate a string.
- str.toUpperCase()
- str.toLowerCase()
- str.trim() // removes whitespaces

strings = immutable
array = mutable

loops - iterable (strings, objects, arrays)

String method
- str.slice(start, end?) // return part of string
- str1.concat(str2) // joins str1 with str2
- str.replace(searchVal, newVal)
- str.charAt(idx)

Arrays. - Non-premetive ~~datatypes~~
collection of items.

looping over an Array
- print all elements of an array.
  for loop : length
  arr[1, 2, 3, 4, 5]
  for (let indx = 0; idx < arr.length; idx++){
  }
- we generally use for-of loop.
  for (let city of cities) {
  console.log(city); }

Arrays method
Push() : Add to end.
Pop() : Delete from end and return.
to string() : convert arrays to strings.
contcat() : joins multiple arrays and return result.
unshift() : add to start
shift() : delete from start and return.
slice() : Return a peice of array
slice (startIdx, endIdx)
splice() : change original array (add, remove, repl
splice (startIdx, delcount, newf(1...)

Functions parameters are like a local variables.
→ They have block scope

Functions :
Block of code that perfroms a specific task. can be invoked whenever needed

call

Function function name() { — function call
}

Function functionName(param1, param2...) {
}

Arrow functions
compact way of writing a function
const functionName = (Param1, param2..) => {
}

const sum = (a, b) => {
return a+b;
}

"Abc".toUppercase — method
string.

Higher order function / Higher order method

ForEach loop in Arrays
arr.ForEach(call back function)
CallbackFunction : Here, it is a function to execute for each element in array.
A callback is a function passed as an argument to another function.

arr.ForEach(val) => {
console.log(val);
})

(For each function is connect with arrays)

(used for return parameter or index)

amstrong, prime, perfect

Some more Array Methods

- Map — new array

Creates a new array with the results of some operation
The value its callback returns are used to from new array
arr.map(callbackFnx(value, index, array))

```
let newArr = arr.map((val) => {
    return val * 2;
})
```

- Filter

Creates a new array of elements that gives true for a condition / Filter.

Eg. all even elements

```
let newArr = arr.filter((val) => {
    return val % 2 === 0;
})
```

- Reduce — If want only one output then we use this function.

Performs some operations and reduces the array to single value. It returns that single value.

Function decelaration
Function without a parameter and return
Function returning value
Function with parameter
Functions with two parameters
Function with many parameters
Function with unlimited number of parameter
- Regual
- arrow

Null is primeative type in javascript
it is a empty object.

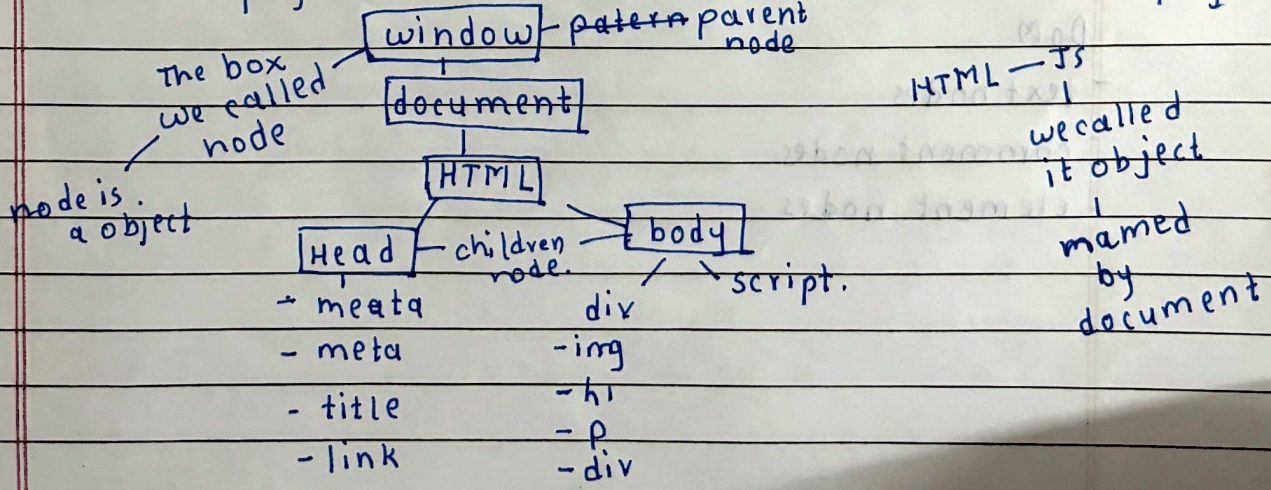
Anonymous Function
Expression function
self inroking function
Arrow function
Function with default parameters
Function declaretion versus Arrow function.

-DOM- Document object Model
3 musketeers of web development
HTML- structure
CSS - style
JS - logic

-<style> tag connects HTML with CSS <link rel = " " href = " "> ← Its good to add this in head
-<script> tag connects HTML with JS <src = " " ← Its good to add this in body at last.

Window object
The window object represents an open window in a browser.
It is browser object (non-javascript's) and is automatically created by browser.
It is a global object with lot of properties and methods

DOM-
When web page is loaded. the browser creates DOM of the page.



console.log - print

console.dis - document - properties
is a model object. \ method

DOM Manipulation

- selecting with id - id is unique for each.
document.getElementById("myId")

- selecting with class - HTML collection - similar to array
document.getElementByClassName("myClass")

- selecting with tag
document.getElementByTagName("p")

- query selector — Node list)
document.querySelector("myId/myclass/tag")
// return First element
document.querySelectorAll("myId/myclass/tag")
// returns a Nodelist

Properties - —value can be change get, set.

- tagName : returns tag for element nodes
- innertext : returns the text content of the element and its children
- innerHTML : returns the plain text or HTML contents in the element
- textcontent : returns textual content even for hidden elem

DOM

- text nodes
- comment nodes
- element nodes

- Attributes

getAttributes(attr) // to get the attribute value.
setAttribute(attr, value) // to set the attribute val th

- style

node.style

- Insert elements - let el = ~~elements~~ document.createElement("div")

node.append(el) // adds at the end of node (inside)
node.prepend(el) - adds at the start of node (inside)
node.before(el) - adds before the node (outside)
node.affer(el) - adds after the node (outside)

Dele~~me~~te element

- node.remove() - remove the node

Events in Javascript.

The change in the state of an object is known as events.
Events are fired to notify code of "intresting changes" thayt may affect code execaution.
Mouse events (click, double click etc.)
Keyboard events (keypress, keyup, keydown).
From events (subit etc).
Print event and many more.

Event handling in JS

node.~~Js~~event = () => {
    // handle event }

If object and prototype have same method object method will be used

- array is a object.

- Event object

It is a special object that has detailed about the event
All event handlers have access to the Event object's properties and methods

```
node.event = (e) => {
    // ... }
```

e.target, e.type, e.clientX, e.clientY

- Event listensers

callback reference should same to remove

node.addEventListener(event, callback)

node.removeEventListener(event callback) — also handler. It is function

Object oriented programming
classes and objects

- Prototypes in JS

It is object

A javascript object is an entity having state and behavior (properties and method)

reference to an object

Js object have special type (property) called prototype
We can set prototype using __proto__

- classes — help to making object template or blueprint

class is a program-code template for creating objects. Those objects will have some sate (variables) and some behaviour (functions) inside it

```
class MyClass {
  constructor(){...}
  myMethod(){...}
}

let myObj = new MyClass{...}
```

constructor() method is:
- automatically invoked by new
- initializes object

```
class Myclass {
  constructor() { ... }
  myMethod() { ... }
}
```

- Inheritance

inheritance is a passing down properties and methods From parent class to childelass. - (extend it for class 1 to calss 2)

```
class parent {
  }
class child extends Parent {
  }
```

* If child and parent have same method, child's method will be used. [Method Overriding]

- super keyboard

The super keyboard is used to call the constructor of its parent class to access the parents properties and methods.

```
super(args) // call parent's constructors
super.parentMethod(args)
```

async await >> promise chains >> callback hell

Sync

- Synchronous

Synchronous means the code runs in a particular sequence of instructions given in the programm. Each instruction waits For previous instruction to its execution.

Asynchronous

Dute to asynchronous programming sometimes imp instructions get blocked due to some previous instruction which causes a delay in the UI. Asynchronous code execution allows to execute next instructions immediately and doesn't block the flow.

callbacks

A callback is a Function passed as an argument to another function.

callback Hell

Nested callbacks stacked below one another Froming a pyramid structure. (pyramid of Doom)
This style of programming becomes dificult to understand and manage.

Promises

Promise is for 'eventual' completion of task. It is an object It is a solution to callback hell.

let promise = new Promise ((resolve, reject) => { .... })

resolve and rejects are callbacks provided in JS.

function with 2 handlers.

Promise — rejected.
pending / Fullfield (resolved)

Promises objects can be -

state { pending - the result is undefined
Resolved - the result is a value (Fulfilled)
Rejected - the result is an error object

resolve (result)
result. { reject (error)

.then() and .catch()

```
promise.then((res) => {...})
promise.catch((err)) => {...})
```

Async - Await

async function always returns a promise

```
async function myFunc() {...}
```

await pauses the execution of its surrounding async function until the promise is settled.

IIFE : Immediately Invoked Function Expression

IIFE is a function that is called immediately as soon as it is defined.

```
(Function () {
    //....
})();
(() => {
  //...
})();
(async () => {
   //...
})();
```

Fetch API

The Fetch API provides an interface for fetching (sending / receving) resources.

It uses Request and Responces objects

The Fetch() method is used to fetch a resource (data)

```
let promise = fetch (url, [options])
```

↘ get request

AJAX is Asynchronous JS and XML — data format

JSON is Javascript object Notation

json() method: returns a second promise that resolves with result of parsing the responce body text as JSON (Input is JSON), (object is JS object)

Request and Response.

HTTP Verbs — HTTP is Hyper text transfer protocol

Response status code

\haders also contains details about the responses, such as content type HTTP status code.etc.