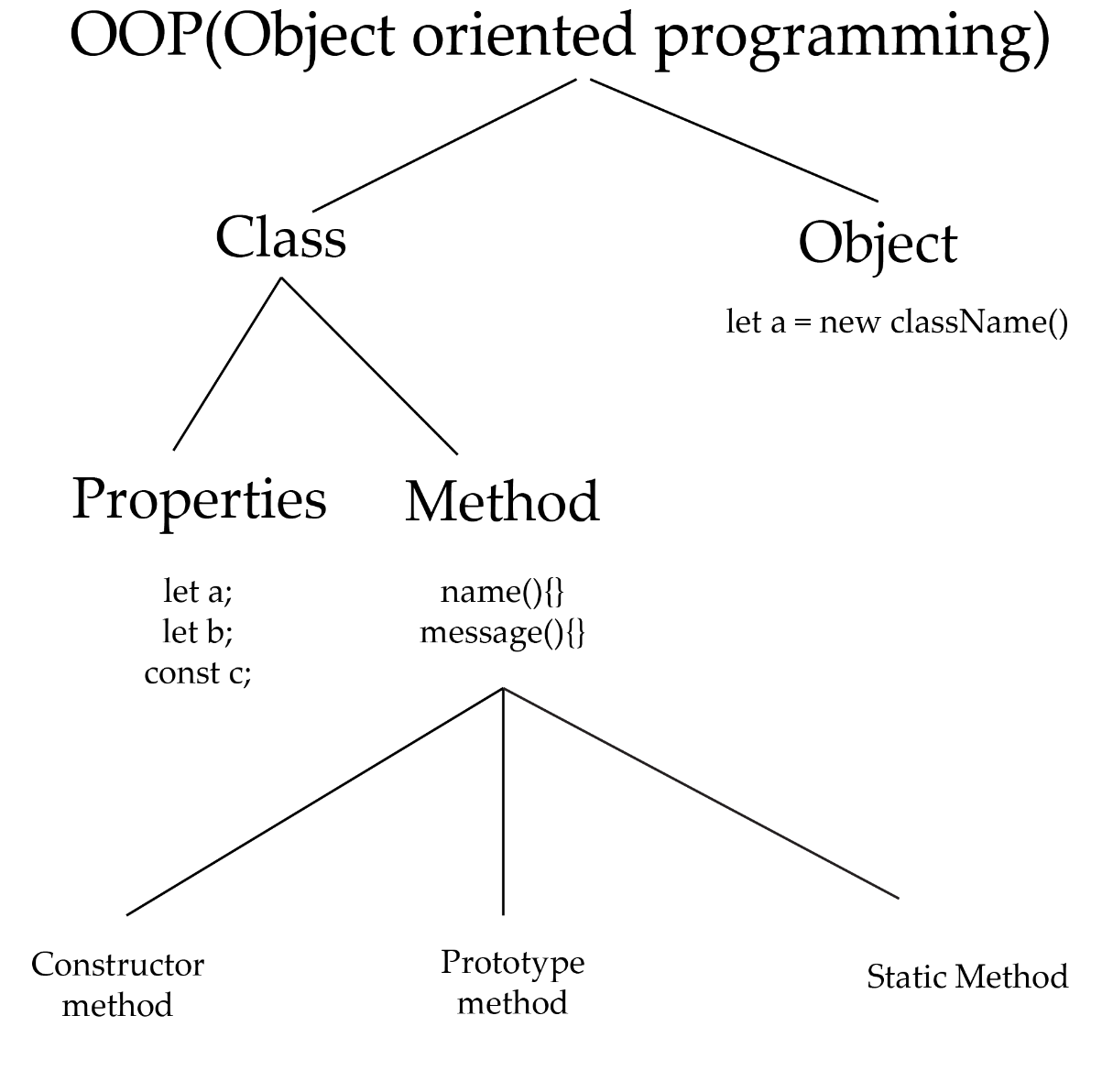
# **OOP**

* OOPS is all about Class and Objects and inheritance
* Everything in this world is an object with some properties, function, methods just like JavaScript considered everything in it are objects
* Class Is like a blue print which holds all things about one individual or personality or type
* Object is like a whole thing holds class, properties
* Inheritance means common properties which class hold



1. **Constructor Method**

* The following example shows the constructor method in class
* Constructor is self-invoking it’s self
* It is use to initialize the objects properties or variables

class student{

  constructor(){

    console.log("its constructor here");

  }

}

let a = new student();

* The following example represent the properties in constructor method
* Send values to constructor by **this**
* **This** represent the class
* Properties can be use by simple variables
* this.studentName and this.studentYear are variable and you can create private variables using # outside the constructor function

class student{

  constructor(name,year){

    this.studentName = name

    this.studentYear = year

    console.log(`student name is ${name} and his admission date is ${year}`);

  }

}

let a = new student("abdullah",2025);

1. Prototype Method

* Simple type of method
* Can called after creating an object
* Here previousMarks() is a prototype method

class student{

  constructor(name,year){

    this.studentName = name

    this.studentYear = year

    console.log(`student name is ${name} and his admission date is ${year}`);

  }

  PreviousMarks(){

    console.log("marks are here");

  }

}

let a = new student("abdullah",2025);

a.PreviousMarks();

* You can put some properties by using 2 ways
* **This** for the properties present in constructor and after calling a method put arguments as a value
* **In constructor you can use variable name because variables are present in it but in prototype or static methods you can only use properties or variables by using this key word**

class student {

   constructor(name, year) {

   this.studentName = name

   this.studentYear = year

 // console.log(`student name is ${name} and his admission date is ${year}`);

      }

      PreviousMarks() {

        console.log(`Student 1 has ${metric} marks in his metric and ${inter} in his inter Examination as well his name is ${this.studentName} and his admission date is ${this.studentYear}`);

      }

    }

    let a = new student("abdullah", 2025);

    let metric = 450;

    let inter = 500

    a.PreviousMarks(metric, inter);

1. Static Method

* Write static before variable name
* Can call **className.method** Name
* No need to create object in this method
* You can’t access variables from constructor even by using **this**
* Can access variables from outer scoop of the class as well from the scoop of the static method
* Due to the above there’s no need to pass value by argument

    class student {

      constructor(name, year) {

        this.studentName = name;

        this.studentYear = year;

      }

      static marks() {

        let markssssss = 400000

        console.log(`Adil \"G\" ${markssssss} + ${kut} `);

      }

    }

    let kut = "kut"

    student.marks();

add values from out side

class Student{

  constructor(name,lastName,city,lastDegree,gender){

    this.stuName = name;

    this.stuLastName = lastName;

    this.stuCity = city;

    this.stuLastDegree = lastDegree;

    this.stuGender = gender;

    console.log(this.aa);

  }

  info(){

    let basicSalary = 3000;

    let incentive = 1200;

    let mobileAllowns = 2000;

    let totalSalary = basicSalary + incentive + mobileAllowns

    console.log(` ${this.aa}`);

  }

}

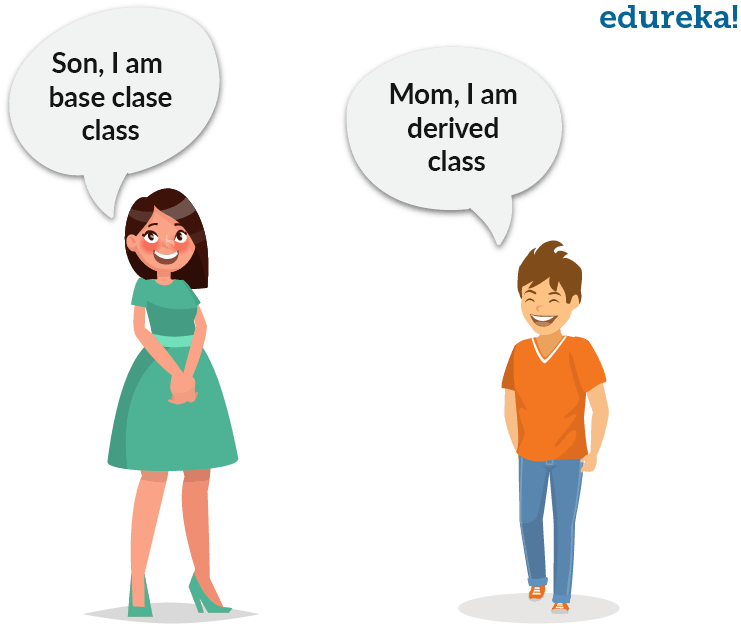
Student.prototype.aa = "harammoo"

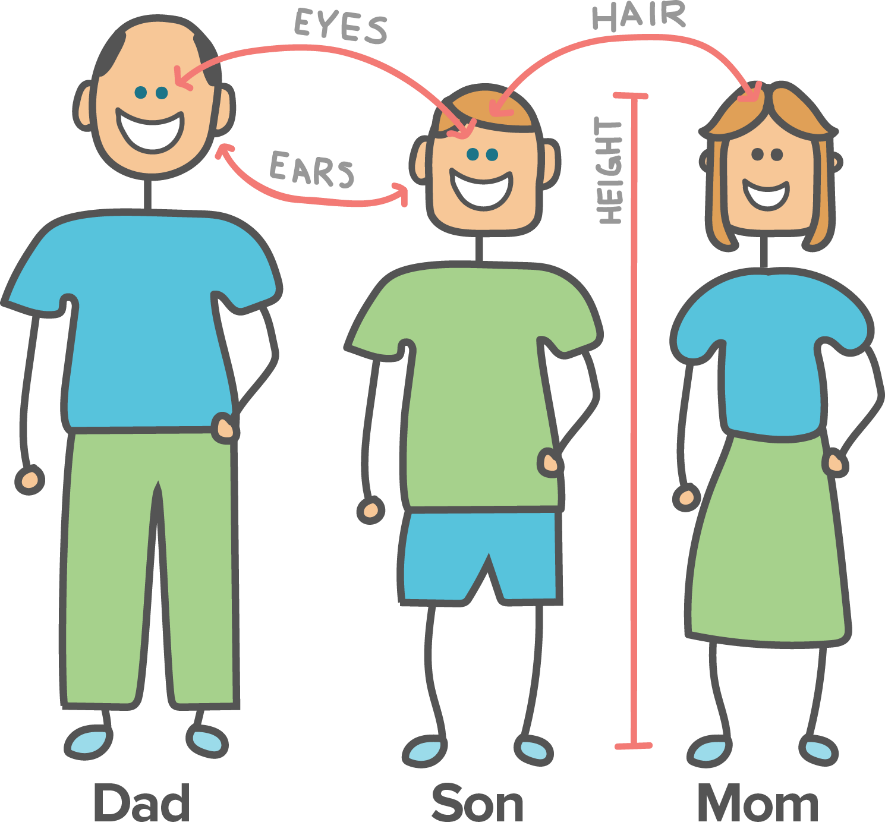
let stuObj = new Student("adil","younas","gujranwala","Msc","male")

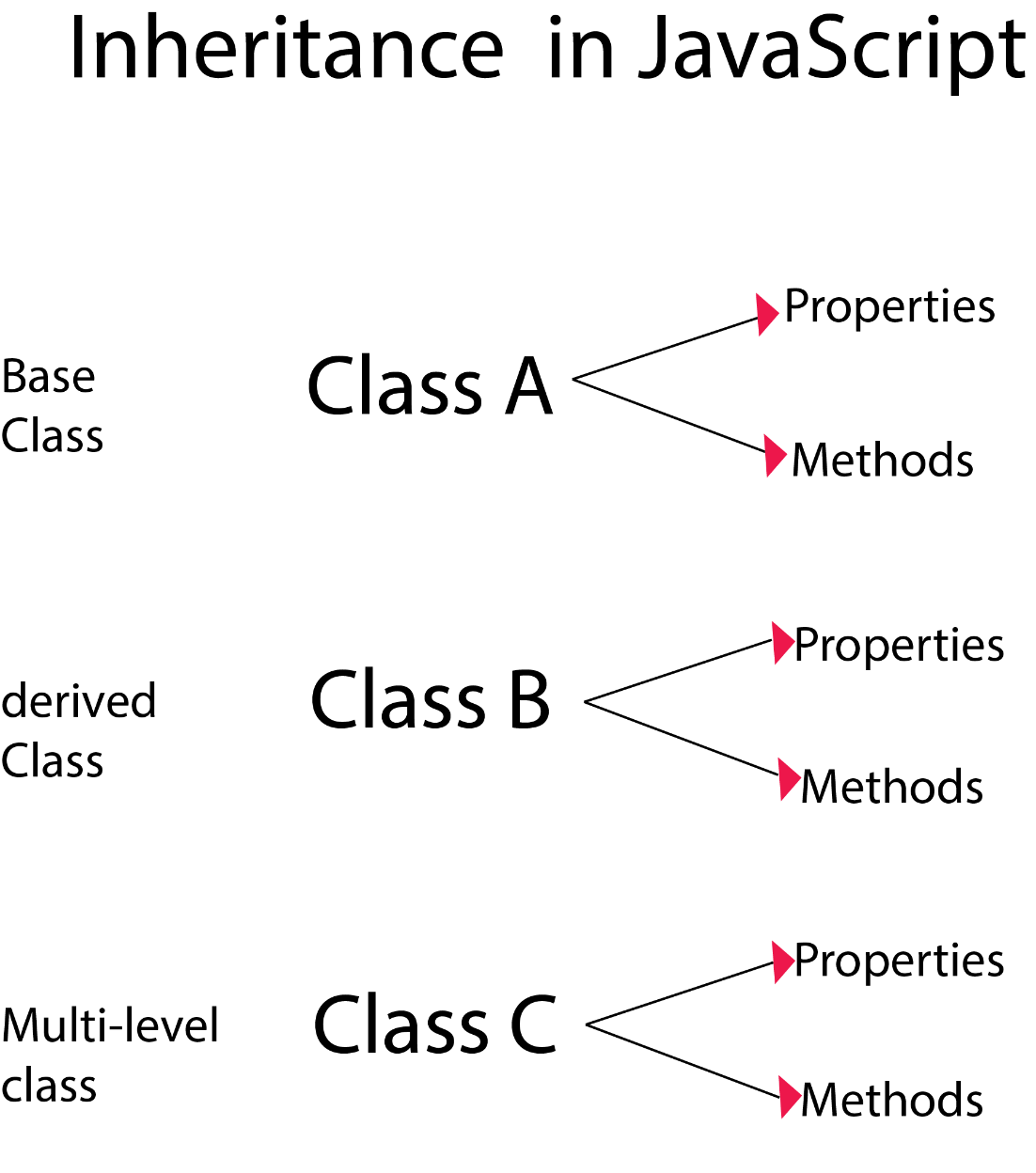
stuObj.info()

1. Inheritance

* Inheritance mean that the derived class have properties of its base class
* Father called base class and son called derived class







* Put value as an argument in object must delivered to constructor present in base class is a database of all derived class
* If Extends class is empty then it shows base class content

  constructor(){

    console.log("employee");

  }

}

class manager extends employee{

}

let a = new manager()

* Extends have attitude run it’s value first if there is 2 constructor present in it, it will throw error of required super()

class employee{

  constructor(){

    console.log("employee");

  }

}

class manager extends employee{

  constructor(){

    console.log("manager");

  }

}

let a = new manager()  //Uncaught ReferenceError: Must call super constructor in derived class before accessing 'this' or returning from derived constructor

class employee{

  constructor(){

    console.log("employee");

  }

}

class manager extends employee{

  constructor(){

    super() //super() is required

    console.log("manager");

  }

}

let a = new manager()

* If there are two prototypes then it’s ok then.
* If you want to display data then call super.info()

class employee{

  constructor(){

    console.log("employee");

  }

  info(){

    console.log("employee info");

  }

}

class manager extends employee{

  constructor(){

    super()

    console.log("manager");

  }

  info(){

super.info()

    console.log("manager info");

  }

}

let a = new manager()

a.info()

**In Static method**

* **Static method just behaves like prototype in inheritance**

class employee{

  static infoo(){

    console.log(" e static");

  }

}

class manager extends employee{

  static infoo(){

    super.infoo()

    console.log(" m static");

  }

}

let a = new manager()

manager.infoo()

**Real life Example of inheritance**

* **Always declared variables in method**

class employee{

  constructor(name,age,salary){

    this.empName = name

    this.empAge = age

    this.empSalary = salary

    console.log("constructo0r employee");

  }

  info(){

    document.write(`<h3>Employee Class</h3>

                    Name : ${this.empName} <br>

                    age : ${this.empAge} <br>

                    age : ${this.empSalary} <br> `)

  }

}

class manager extends employee{

  info(){

    super.info()

    let ta = 1000;

  let pa = 300;

  let totalSalary = ta + pa + this.empSalary

    document.write(`<h3>Manager Class</h3>

                    Name : ${this.empName} <br>

                    age : ${this.empAge} <br>

                    age : ${totalSalary} <br> `)

  }

}

let a = new manager("Adil",25,500,"Gujranwala",60000);

a.info()

**Multi-level Inheritance**

    class employee {

      constructor(name, age, salary, year) {

        this.empName = name

        this.empAge = age

        this.empSalary = salary

        this.empJoinYear = year

      }

      info() {

        document.write(`<h3>Employee Class</h3>

                         Name: ${this.empName} <br>

                         age: ${this.empAge} <br>

                         slary: ${this.empSalary} <br>

                         joing: ${this.empJoinYear} <br>     `)

      }

    }

    class manager extends employee {

      info() {

        super.info()

        let TA = 2000

        let PA = 1500

        let MA = 1200

        let totalSalary = TA + PA + MA + this.empSalary

        let ManagerName = "Younas Khan";

        let managerJoinYear = 2015

        document.write(`<h3>Manager Class</h3>

                         Name: ${ManagerName} <br>

                         age: ${this.empAge} <br>

                         slary: ${totalSalary} <br>

                         joing: ${managerJoinYear} <br>     `)

      }

    }

    class CEO extends manager {

      info() {

        super.info()

        let TA = 20005

        let PA = 15000

        let MA = 12000

        let totalSalary = TA + PA + MA + this.empSalary

        let CEOName = "Abdullah Warich";

        let CEOJoinYear = 2022

        document.write(`<h3>CEO Class</h3>

                         Name: ${CEOName} <br>

                         age: ${this.empAge} <br>

                         slary: ${totalSalary} <br>

                         joing: ${CEOJoinYear} <br>     `)

      }

    }

    let a = new CEO("Muhammad Adil", 25, 30000, 2023)

    a.info()

# Quick recap

1. Oop generally devided into two parts one is class and other is object
2. Class is like a blue print by using it we create something
3. Class have some properties and methods, properties means variable and methods means function
4. There are three types of methods in oops constructor , prototype, static method

# Informative content

1. Values place as an argument in object will pass in base constructor method but you can use it in below and derived class as well
2. Constructor method store values by this keyword
3. In constructor method after storing the values you can use them by simple variable
4. In prototype method constructor values used by this keyword
5. We are unable to use constructor properties in static method so it can use globle as well as block scoop values

# Inheritance content

1. Always use extends in derived class to use inheritance
2. If there are same constructor methods in base and derived class without super() then it is an error
3. If there are same prototype method as well as static method in base and derived class without super then it’s OK
4. Call one all arrived is called inheritance base has limited basic information but derived has its own properties as well

# To prevent problems and ease to understand must obey

1. Use constructor method in base class only because use can access its values in below as well as in derived class
2. Under condition that you use prototype and static in derived class then use super.info() and simple super() for constructor method
3. Call one prototype and all employees and managers list you create in base and derived class will appear with changes