Classes & Objects

Prototypes in JS

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

==> One way to declare an Object

```
const studentObj = {
    studName: "Hirtik", // State/Property

    rollNo: 210170107030, // State/Property

    marksSPI: 8.20, // State/Property

    //Following is Behaviour/Methods
    printMarkSPI: function () {
        console.log("SPI is ", this.marksSPI);
    }
};
```

JS objects have a special property called prototype. =>

=> Prototype is also an Object.
=> It default in any object and it has some predefined properties/methods.
=> in short, Prototype is an Object of an Object
=> Type of prototype is reference to an Object and NULL in some cases.

We can set prototype using _ _ proto _ _

Explanation:

```
//Following object only defines the properties/methods can employee have.
//First we have employee object and calculateTax function in it and want use employee's properties/Methods in any other object we have to
make prototype of employee object and that will have all the methods and properties.
const employee = {
    calculateTax: function (salary) {
        console.log("Tax is", salary * 0.1); //10% Tax
    }
};

//I am an employee, Hirtik!
//I just have my property salary.
const employeeHirtik = {
    salary: 100000,
};

//But I want to calculate tax on my salary. but the method is in employee function. so we can make use of it by setting up employee as an
prototype by using __proto__

employeeHirtik.__proto__ = employee; //This will bring all the methods and properties of employee object and we can use it in
employeeHirtik.

//Calculating tax
employeeHirtik.calculateTax(employeeHirtik.salary);

//We can have many employees we can make use of properties/methods of employee object.

//Extra Example
const employeeHitesh = {
    salary: 90000,
};
employeeHitesh.__proto__ = employee;
employeeHitesh.calculateTax(employeeHitesh.salary);
```

*If object & prototype have same method, object's method will be used.

Classes in JS

Class is a program-code template for creating objects.

Those objects will have some state (variables) & some behaviour (functions) inside it.

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

let myObj = new MyClass( ) ;
```

Classes in JS

Constructor( ) method is :

- automatically invoked by new
- initializes object

```
class MyClass {

   constructor( ) { ... }

   myMethod( ) { ... }

}
```

Inheritance in JS

inheritance is passing down properties & methods from parent class to child class.

```
class Parent {

}

class Child extends Parent {

}
```

*If Child & Parent have same method, child's method will be used. [Method Overriding]

super Keyword

The super keyword is used to call the constructor of its parent class to access the parent's properties and methods.

```
super( args )  // calls Parent's constructor

super.parentMethod( args )
```

Let's Practice

Qs. You are creating a website for your college. Create a class User with 2 properties, name & email. It also has a method called viewData( ) that allows user to view website data.

Qs. Create a new class called Admin which inherits from User. Add a new method called editData to Admin that allows it to edit website data.

Error Handling

try-catch

```
try {
    ... normal code
} catch ( err ) { //err is error object
    ... handling error
}
```