Object oriented programing

Data Types

There are two types of data **primitive** and **non-primitive**

1. Primitive Data type

Numbers, Boolean, String, Null, Undefined

* Immutable (we can’t add or delete values from them like object and array)

Like let a = 100

a.100 // we can’t do that

* Pass by value (when you store them in a variable)

let a = 40

let b = a //b = 40

//pass by value like b = 40 now if you change a it

//will not put effect on a

console.log(a); //40

console.log(b); //40

a = 11111

console.log(a); //11111

console.log(b); //40

1. Non-Primitive data type

Array and objects are the non-primitive

* Mutable (can add values here)
* Pass by reference (change will reflect both sides)

const a = {

    name:"Adil",

    age:25,

}

const b = a

console.log(a);  //{name: 'Adil', age: 25}

console.log(b); //{name: 'Adil', age: 25}

b.father = "younas"

delete b.age

console.log(a); //{name: 'Adil', father: 'younas'}

console.log(b); //{name: 'Adil', father: 'younas'}

Two ways to add function in object

const a = {

  name: "Adil",

  age: 25,

  info: function info() {

    console.log("adil younas");

  },

};

a.info(); //adil younas

const b = {

  name: "Adil",

  age: 25,

  info: () => {

    console.log("adil younas");

  },

};

b.info(); //adil younas

Object Literal / Simple Object

* Use to store single type of data
* For simple use only
* Can’t be productive if there are many players or single type of object data

const RomanReigns = {

    weigh:255,

    height:150,

}

//what is you have 60 WWE player

//than it is not a productive aproach

Factory Function (return object in key value)

* For normal routine
* Productive In case you have many WWE player but not so many
* Small cap function name only
* Must return something so we don’t use new
* this can’t be use here

//factory function

function wwe(a, b, c) {

  return {

    weight: a,

    height: b,

    group: c,

  };

}

const RomanReign = wwe(2, 4, "B+"); //it is a object

const SathRolline = wwe(4, 40, "C"); //it is a object

console.log(RomanReign); //{weight: 2, height: 4, group: 'B+'}

console.log(SathRolline); //{weight: 4, height: 40, group: 'C'}

abi tak key value pair chal raha tha

Constructor Function (can’t use constructor function here so we use class)

above we use key value pair and here we use this

Here we can’t use function so we use classes

* Capital case function name only
* No need to return
* Because of no return we use new
* this.weight and this.height create a variable inside the function and store a and b in it frequently.
* We can’t create a function inside it because it will not work properly.

function WWE(a, b, c) {

  this.weight = a;

  this.height = b;

  this.group = c;

}

const RomanReign = new WWE(2, 3, "B+"); //it is a object

const SethRolline = new WWE(21, 31, "C"); //it is a object

console.log(RomanReign);

//{weight: 2, height: 3, group: 'Group'}

console.log(SethRolline);

//{weight: 21, height: 31, group: 'C'}

Tip : Constructor function on industrial level

* we can push into array passing through for loop

function WWE(a, b, c) {

  this.weight = a;

  this.height = b;

  this.group = c;

}

const RomanReign = new WWE(2, 3, "B+"); //it is a object

const enemies = []

for (let i = 0; i < 100; i++) {

    enemies.push(new WWE(Math.floor(Math.random()\*100) , 40 ,"B+" ))

}

console.log(enemies); //100 objects in an Array

//(100) [WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE, WWE]

Note: Boletes about class is the simplest way to understand the class they are written by practice

Note 2: Class is nothing but create a variable and pass it value

Class Or Relate Object To Class (this is used here) …

* it is a object but when you console it shows class name with object like className { }
* you can initialize variable without let, const, var and through constructor function or any function can assign the variable a value
* through **#**  you can create a private variable and private function which can’t be access by **dot**
* this create variable if there is no one and if there is then assign it value

1. **constructor function get value by this.**

class Star{

    constructor(size,weight){

        this.a = size

        this.b = weight

    }

}

const RomanReigns = new Star(21,55)

console.log(RomanReigns); //Star {a: 21, b: 55}

1. **any function get value by this**

class Star {

  info(a, b) {

    console.log(a, b);

  }

}

const RomanReigns = new Star(21, 55);

RomanReigns.info(40, 44); //40 44

{…………….……….Inheritance in Classes…….…..………}

Note: In inheritance you can use the variables and function of parent class but you must have to send values

Note 2: super() is constructor of parent class. Receive values from child object and send it to the parent class and then use it

class Electronics {

  #input;

  #output;

  #name;

  constructor(input, output, name) {

    this.#input = input;

    this.#output = output;

    this.#name = name;

  }

  print() {

    console.log(this.#input, this.#output, this.#name);

  }

}

const electronics = new Electronics(12, 51, 24, "140w");

console.log(electronics);

//step 2

class Charger extends Electronics {

  #watt;

  constructor(input, output, name, watt) {

    super(input, output, name);

    this.#watt = watt;

  }

}

const charger = new Charger(12, 51, 24, "140w");

console.log(charger);

//Charger {#input: 12, #output: 51, #name: 24, #watt: '140w'}