Step 1: npm init

Step 2: Đặt tên của project và các thông tin cần thiết để lưu vào file package.json.

Step3: gõ ***npm install –save es6-module-loader traceur*** để cài đặt traceur.

Step 4: gõ ***npm install –save-dev lite-serve*** để cài đặt lite server.

Step 5: Vào file package.json thêm vào dòng **“dev”: “lite-server”** để run lite server.

{

*"name"*: "drones",

*"version"*: "1.0.0",

*"description"*: "",

*"main"*: "index.js",

*"scripts"*: {

*"test"*: "echo \"Error: no test specified\" && exit 1",

*"dev"*: "lite-server"

},

*"author"*: "",

*"license"*: "ISC",

*"dependencies"*: {

*"es6-module-loader"*: "^0.17.11",

*"traceur"*: "0.0.111"

},

*"devDependencies"*: {

*"lite-server"*: "^2.4.0"

}

}

Step 6: gõ ***npm run dev*** để thực thi.

Ở ES6 ta có strict mode, là nếu không khai báo biến mà dùng thì báo lỗi ngay.

Ta hãy bắt đầu với một số dòng code đơn giản.

**Class Creation**

*class* Drone {

}

*let* drone = new Drone();

*console*.log(typeof Drone);

// Result: function

*console*.log(typeof drone);

// Result: object

*console*.log(drone instanceof Drone);

// Result: true

**Constructors and Properties**

*class* Drone1 {

// Phương thức khởi tạo mặc định không tham số của một class.

*constructor*() {

*console*.log('in Drone constructor');

}

// Phương thức khởi tạo có một đối số.

*constructor*(*id*) {

*console*.log('id: ' + id);

}

// Phương thức khởi tạo có 2 đối số.

*constructor*(*id*, *name*) {

*console*.log('Id: ' + id + ' name: ' + name);

}

}

// let drone = new Drone1('A1234');

*class* Drone {

// Để giữa cho constructor đơn giản và dễ test ta có thể làm như sau

*constructor*(*id*, *name*) {

this.id = id;

this.name = name;

}

}

*let* drone = new Drone('A123', 'Flyer');

*console*.log('drone: ' + drone.id + ' ' + drone.name);

// Hoặc cũng có thể truy cập bằng cách này.

*console*.log('drone: ' + drone['id'] + ' ' + drone['name']);

**Static Propterties.**

*class* Drone {

*constructor*(*id*, *name*) {

this.id = id;

this.name = name;

}

}

Drone.maxHeight = 20000;

*let* drone = new Drone('A123', 'Flyer');

*console*.log(Drone.maxHeight);

// Result: 20000

*console*.log(drone.maxHeight);

// Result: undefined.

// Lý do là vì maxHieght không nằm trong instance properties.

// instance properties là những properties khai báo ngay bên trong class đó

// Chứ không phải là thêm vào từ bên ngoài.

// maxHeight bên trên được gọi là static properties hay class properties.

**Method**

*class* Drone {

*constructor*(*id*, *name*) {

this.id = id;

this.name = name;

}

fly() {

*console*.log('Drone ' + this.id + ' is flying');

// Nếu ta quên từ khóa this trước id (this.id) thì

// sẽ có lỗi ngay lập tức vì nó méo biết là id này là

// thằng méo nào, còn this.id là propertiy của class này.

}

}

*let* drone = new Drone('A123', 'Flyer');

drone.fly();

// Result: Drone A123 is flying

**Static methods.**

*class* Drone {

*constructor*(*id*, *name*) {

this.id = id;

this.name = name;

}

static getCompany() {

*console*.log('in getCompany');

}

static getCompany1() {

// Báo lỗi vì một static method

// không thể truy cập đến mấy cái instance properties

*console*.log(this.id);

}

fly() {

*console*.log('Drone ' + this.id + ' is flying');

}

}

*let* drone = new Drone('A123', 'Flyer');

Drone.getCompany();

// Result: in getCompany

//drone.getCompany();

// Báo lỗi vì chỉ có thằng cha mới truy cập được,

// còn mấy thằng con new từ thằng cha thì không truy cập được.

Drone.getCompany1();

// Result: undefined

// Vì một static method thì không thể access

// bởi instance variable hay instance properties.

**Getters and setters**

*class* Drone {

*constructor*(*id*) {

this.\_id = id;

}

*get* id() {

// Ta có thể làm một mớ thứ trong này trước khi lấy giá trị ra

return this.\_id;

}

*set* id(*value*) {

// Trước khi gán value ta có thể làm gì đó cũng được

// Như validation hay manipuilate giá trị đó.

this.\_id = value;

}

}

*let* drone = new Drone('A123');

*console*.log(drone.id);

// Result: A123

drone.id = 'B456';

*console*.log(drone.id);

// Result: B456

**Inheriting.**

**Khi một class kế thừa từ một class thì nó phải bắt buộc gọi đến phương thức khởi tạo của class cha (dùng từ khóa super)**

*class* Vehicle {

*constructor*() {

*console*.log('constructing Vehicle');

}

}

*class* Drone extends *Vehicle* {

}

*class* Car extends *Vehicle* {

*constructor*() {

super();

*console*.log('constructing car');

}

}

*let* c = new Car();

// Kết quả là constructing Vehicle và constructing car.

**Gọi phương thức khởi tạo 1 đối số từ lớp con lên lớp cha.**

*class* Vehicle {

*constructor*(*licenseNum*) {

this.licenseNum = licenseNum;

}

}

*class* Drone extends *Vehicle* {

}

*class* Car extends *Vehicle* {

*constructor*(*licenseNum*) {

super(licenseNum);

}

}

*let* c = new Car('A123');

*console*.log(c.licenseNum);

// Kết quả là A123

**Inhreting Properties.**

*class* Vehicle {

*constructor*() {

this.gpsEnabled = true;

}

}

*class* Drone extends *Vehicle* {

}

*class* Car extends *Vehicle* {

*constructor*() {

super();

this.gpsEnabled = false;

}

}

*let* c = new Car();

*console*.log(c.gpsEnabled);

// Result: false

*class* Vehicle {

start() {

*console*.log('starting vehicle');

}

}

*class* Car extends *Vehicle* {

}

*let* c = new Car();

c.start();

// Kết quả là starting vehicle.

*class* Vehicle {

start() {

*console*.log('starting vehicle');

}

}

*class* Car extends *Vehicle* {

start() {

*console*.log('starting car');

}

}

*let* c = new Car();

c.start();

// Kết quả là starting car

// Vì start của vehicle đã bị start của car override.

*class* Vehicle {

start() {

*console*.log('starting vehicle');

}

}

*class* Car extends *Vehicle* {

start() {

// Đặt từ khóa super trên hay dưới điều không báo lỗi

// Nhưng kết quả sẽ thay đổi thứ thự vì nó sẽ gọi

// cha trước hoặc gọi con trước.

super.start();

*console*.log('starting car');

}

}

*let* c = new Car();

c.start();

// Result: starting vehicle và starting car.

*class* Vehicle {

start() {

*console*.log('starting vehicle');

}

static getCompanyname() {

*console*.log('My Company');

}

}

*class* Car extends *Vehicle* {

start() {

super.start();

*console*.log('starting car');

}

}

*let* c = new Car();

c.start();

Car.getCompanyname();

// Result: My company, lớp con truy xuất được một static method ở lớp cha.

*class* Vehicle {

start() {

*console*.log('starting vehicle');

}

static getCompanyname() {

*console*.log('My Company');

}

}

*class* Car extends *Vehicle* {

start() {

super.start();

*console*.log('starting car');

}

static getCompanyname() {

*console*.log('My Other Company');

}

}

*let* c = new Car();

c.start();

Car.getCompanyname();

// Result: My Other Company, lớp con có thể override lại một static method từ lớp cha

*class* Vehicle {

start() {

*console*.log('starting vehicle');

}

static getCompanyname() {

*console*.log('My Company');

}

}

*class* Car extends *Vehicle* {

start() {

super.start();

*console*.log('starting car');

}

static getCompanyname() {

super.getCompanyname();

*console*.log('My Other Company');

}

}

*let* c = new Car();

c.start();

Car.getCompanyname();

// Result: My Other Company và My Other Company

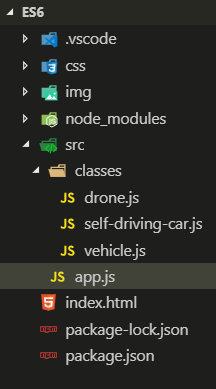
// lớp con có thể override lại một static method từ lớp cha và gọi luôn cả cha lẫn nó.

// Lưu ý là nếu c.getCompanyname() thì sẽ báo lỗi ngay lập tức vì static chỉ có thằng

cha truy cập mà thôi.

**Organizing Classes into Files**

**Tạo cấu trúc file và thư mục như hình sau:**



**Trong app.js**

import {

SelfDrivingCar

} from './classes/self-driving-car.js';

import {

Drone

} from './classes/drone.js';

import {

Vehicle

} from './classes/vehicle.js';

*let* c = new SelfDrivingCar();

*let* d = new Drone();

// Ta sẽ bị lỗi ngay lập tức nếu cố gắn tạo một đối tượng từ Vehicle.

//let v = new Vehicle();

*console*.log(c);

*console*.log(d);

**trong self-driving-car.js.**

import {

Vehicle

} from './vehicle.js';

export *class* SelfDrivingCar extends *Vehicle* {

}

**Trong drone.js**

import {

Vehicle

} from './vehicle.js';

export *class* Drone extends *Vehicle* {

}

**Và trong vehicle.js**

export *class* Vehicle {

}

**Loading data**