**Chương 1: Arrow Function**

Mô tả:

// hàm arrow function

Const logger = (log) => {

Console.log(log)

}

Logger(“hello world”)

//hàm tổng

Const sum = (a,b) => {

Return a+b;

}

Console.log(sum(2,2));

==>

Const sum = (a,b) => a+b

* Tự hiển là return luôn nếu ko có dấu “{}”

//hàm rút gọn

Const sum = (a,b) => ({a:a,b:b});

//

Const course = {

Name: “javascript basic!”,

//arrow function ko có context

getName: () => {

return this;

}

}

//constructor

Const Course = (name,price) => {

This.name = name,

This.price = price

}

=> là constructor ko thể sử dụng arrow function

**Chương 2: Module trong javascript**

Khái niệm:

* Import./Export

Import:

* Import logger,{TYPE\_LOG} from “./logger.js”;
* Import {TYPE\_LOG} from “./constants.js”;
* Import \* as constants from “./constans.js”;
* Import {default} from ‘./looger.js’;
* Import {default as logger2} ‘./logger/index.js’;

Export:

Function logger(log,type=’log’) {

Console[type](log);

}

Export default logger;

// export ra biến

Export const TYPE\_LOG = ‘LOG’;

**Chương 3 Enhanced object literals trong javascript**

Cách khai báo:

cách 1:

Var name = ‘’;

Var price = 1000;

Var course = {

Name,

Price,

Getname() {

Return name;

}

}

Console.log(course);

Cách 2

Var fieldname = ‘’name’;

Var fieldprice = ‘pirce’;

Const course = {

[fieldname]: ‘dungct’,

[fieldprice]: 1000

}

Console.log(course);

**Chương 4: spread trong javascript**

Toán tử REST:

Cách 1:

Function logger(a,b, … params) {

Console.log(params);

}

* a,b sẽ là 1,2 và params sẽ là phần còn lại

Cách 2:

//Khai báo thông thường:

Function logger(obj) {

Console.log(obj.name);

console.log(obj.price);

}

//Khai báo khác

Function logger({name, price, …rest}) {

Console.log(name);

Console.log(price);

}

//object

Logger ({

Name:’javascript’,

Price:100,

Description:’none’

})

// dang mảng

Logger([2,6,12])

Function logger([a,b,…rest]) {

Console.log(a);

Console.log(b);

Console.log(rest);

}

Toán tử spread:

// cách thông thường

Var array 1 = [“1”,”2”];

Var array 2 = [“3”,”4”];

Var array3 = […array1,…array2];

// đối với object

Var object1 = {

Name: ‘javascript’

}

Var object2 = {

Price: 1000

}

Var object3 = {

…object1,

…object2

}

Console.log(object2);

// overide ghi đè mảng

Var defaultconfig = {

Api: ‘http://’,

Apiversion: ‘v1’,

Other: ‘other’

}

Var execiseconfig = {

…default,

Api: ‘http://localhost’

}

//

Var array = [‘rudy’,’php’,’react’];

Function loogerr(…rest) {

For(var i=0; i<rest.legth, i++) {

Console.log(rest[i]);

}

};

Logger(…array);

**Chương 5: Destructuring trong javascript**

Cách 1:

Var array = [‘javascript’,’php’,’ruby’];

Var [a,b,c] = array;

Console.log(array);

Cách 2:

//Object

Var course = {

Name: ‘dungct’,

Price: 1000,

Chilldren: {

Name: ‘reactjs’

}

}

Var {name: Pareentname, children: {name: childrenName}} = course

Console.log(parentName);

Console.log(childrenName);

Cách 3:

Var {name, descripttion = ‘none’} =course

Console.log(description);  
🡺 nếu trong object có thằng description dc gán 1 giá trị thì nó sẽ lấy ra còn ko có thì lấy value

//Toán tử rest trả về mảng giống như console.log

Function logger(…param) {

Console.log(param);

}

Console.log(logger(1,2,3));

**Chương 6: Document.createElement()**

Khai báo:

//

Const h1 = document.querySelector(‘#heading’);

Console.dir(h1);

**//**

**Const h1 = document.creatElement(‘h1’)**

**h1.innerText = ‘hello guys’**

**h1.id = ‘heading’**

**h1.classname = ‘test class-2’**

**h1.style.color = ‘red’**

**console.log(h1);**

**document.body.appendChild(h1)**

**//**

**Const root= document.getElementById(‘root’)**

**Object.assign(h1.style,{**

**Color: ‘blue’,**

**backgroundColor: ‘#333’**

**})**

**Root.appendChild(h1);**

**Chương 7: Github,NPMJS,UNPKG**