*Instruction JavaScript*

1, 2, 4: bật lên cửa sổ hộp thoại.

1. Alert : hiện lên thông báo
2. Console: bảng điều khiển để truy xuất thông tin khi chương trình chạy

Ex:

console.log();

console.error();

console.warn();

console.time();

console.timeEnd();

console.group();

console.groupEnd();

console.clear();

1. Confirm: hiện thông báo oke or cancel
2. Prompt: hiện thông báo và nhập kí tự
3. Settimeout(function(){}, time): thực thi đoạn code sau khoảng time nhất định
4. Set interval(function(){}, time) : thực thi liên tục trong 1 khoảng time nhất định
5. ***Kiểu dữ liệu và toán hạng***

/\*

    Các toán hạng: +,  cộng -,  trừ \*,  nhân /,  chia %,  mod

        ++, += 1 --, -=1 \*\*, mũ

        ++a:

            việc 1:  + 1 cho a => a = a + 1;

            việc 2:  trả về a sau khi được + 1

        a++:

            việc 1: "a copy"

            việc 2: cộng 1 của a => a = a - 1;

            việc 3: trả về "a copy"

    Boolean: wrong: 0, false, "", '', undefined, NaN, null

    Symbol: var id = Symbol('id');  // Unique: duy nhất

    ==: chỉ so sánh giá trị

    ====: so sánh kiểu dữ liệu

    &&: lấy giá trị cuối != wrong

    ||: lấy giá trị đầu != wrong

Các giá trị falsy: a **(chuỗi rỗng)**, b **(số 0)**, d **(false)**

Các giá trị truthy: c **(mảng rỗng)**, e **(object rỗng)**, f **(chuỗi)**

\*/

// Number type

// String type

// Boolean type

// Undifined type

// Null

// Symbol

// Function

1. ***STRING***

// Tạo chuỗi: có 2 cách nên sử dụng cách 1

// c1:

// var fullName = "Vuong Duc Trong";

// var fullName = new String("Vuong Duc Trong");

// 1 số case sử dụng backslash(\)

// 2 dấu nhập vào "\\" = \ trên màn hình

// var fullName = 'Vuong Duc Trong la \'Sieu Nhan\'';

// var firstName = 'Trong';

// var LastName = 'Vuong';

// c1

// console.log('Toi la ' + firstName + ' ' + LastName);

// c2

// console.log(`Toi la ${firstName} ${LastName}`);

/\*

    Thuộc tính của String:

        1. s.length: độ dài chuỗi

        2. s.indexOf('kí tự trong chuỗi s'): vị trí đầu của chuỗi kí tự trong s.

                + in ra vị trí bắt đầu của kí tự

                + nếu không có in ra -1

           s.LastIndexOf('kí tự trong chuỗi s'): vị trí cuối của chuỗi kí tự trong s.

        4. s.search('kí tự trong chuỗi s'): tìm vị trí của chuỗi kí tự trong s.

        5. s.slice(start index, end index): trích chuỗi con trong s.

            có thể cắt 2 hướng trái phải, phải trái:

                ví dụ: [2, 5] or [-3, -1]

        6. s.replace('kí tự trong chuỗi s', 'kí tự mới'): thay thế chuỗi kí tự đầu tiên trong s.

           s.replace(/ kí tự trong chuỗi s /g , 'Kí tự mới'): thay thế toàn bộ chuỗi kí tự trong s.

        7. s.toUpperCase(): chuyển sang chữ hoa.

           s.toLowerCase(): chuyển sang chữ thường.

8. s.trim(): loại bỏ khoảng trắng đầu và cuối xâu s

        9. s.split(', '): cắt chuỗi string thành Array

            ex: s = 'java, C++, Python' => s = ["java", "C++", "Python"]

            s.split(','): => s = ["java", " C++", " Python"]

            s.split(''); => s = ["j", "a", "v", "a"]

        10. s.charAt(index): kí tự ở vị trí index - 1: nếu index >= s.length(): => ''

            s[index]: kí tự ở vị trí index - 1: nếu index >= s.length(): => undefined

 11. number.toString(): chuyển số thành xâu

        12. number.toFixed(): làm tròn số và chuyển số thành xâu

            number.toFixed(a): làm tròn đến số thập phân thứ a

1. ***ARRAY***

\*/

var languages = [

    'JavaScript',

    'PHP',

    'Ruby',

    'Dart'

];

// languages.isArray(): kiểm tra Array

// languages.length: độ dài Array

// languages[i]: lấy giá trị trong Array

// 1. s,ToString(): chuyển Array sang chuỗi

// 2. S.Join(‘ - ’): thêm kí tự cho từng chuỗi trong xâu

// 3. Pop: xóa giá trị cuối và in ra giá trị đó : rỗng in ra: undefined

console.log(languages.pop());

// 4. Push: thêm 1 or nhiều phần tử phần tử vào cuối mảng và trả về độ dài mảng

console.log(languages.push('Dart'));

// 5. Shift: xóa phần tử đầu mảng và in ra phần tử đó : rỗng in ra: undefined

console.log(languages.shift());

// 6. Unshift: thêm 1 or nhiều phần tử phần tử vào đầu mảng và trả về độ dài mảng

console.log(languages.unshift('JavaScript'));

// 7. Splicing: xóa or chèn

// languages.splice(start, delete\_count); xóa delete\_count phần tử bắt đầu từ vị trí start

languages.splice(1, 1);

// languages.splice(1, 0, 'Dart'); xóa 0 phần tử bắt đầu từ 1 và chèn 'Dart' vào vị trí 1

languages.splice(1, 0, 'Dart');

console.log(languages);

// 8. Concat: nối Array

var languages2 = [

    'C++',

    'Python',

];

console.log(languages.concat(languages2));

// 9. Slicing: lấy ra mảng con từ st -> end, xuôi ngược đều được. or copy mảng

// cắt từ index = [start, end)

console.log(languages.slice(1, 2));

// index = [0) copy Array ban đầu

console.log(languages.slice(0));

/\*

***Array Method:***

        forEach(): in ra danh sách Array

        every(): & tất cả đk thỏa mãn in ra true

        some(): || chỉ cần 1 đk thỏa mãn in ra true

        find(): tìm kiếm trả về 1ptu or undefined khi 0 có

        filter(): tìm kiếm trả về tất cả phần tử or [] khi 0 có

// 1. dễ hiểu, 2. ngắn gọn, 3. hiệu năng

function getMostFavoriteSport(array) {

    var result = array.filter(function(sports) {

        return sports.like > 5;

    });

    return result;

}

**map()**: thêm được các thuộc tính <key, value>

1.Lưu trữ các cặp khóa-giá trị:

2.Duy trì thứ tự chèn:

3.Kích thước động:

4.Các phương thức tiện lợi:

var courses = ['HTML & CSS', 'Javascript', 'PHP', 'Java']

function render(courses) {

    var html = courses.map(function(value) {

        return '<li>' + value;

    }).join('</li>');

    var htmlElement = document.querySelector('ul');

    return htmlElement.innerHTML = html;

}

**Ví dụ: Với mảng var courses = ['ReactJS', 'AngularJS', 'VueJS'] sẽ thu được kết quả:**

* **ReactJS**
* **AngularJS**
* **VueJS**

**reduce():** tính tổng

function getTotalGold(array) {

    var sum = array.reduce((sum, current) => sum + current.gold, 0);

    return sum;

}

// Expected results:

var arr = [

    ['name', 'Sơn Đặng'],

    ['age', 18],

];

function arrToObj(array) {

    return array.reduce((object, [key, value]) => {

            object[key] = value;

            return object;

        }, {})

}

console.log(arrToObj(arr)); // { name: 'Sơn Đặng', age: 18 }

\*/

**Filter :** tạo phương thức myFilter

Array.prototype.myFilter = function(cb) {

    var output = [];

    for(var index in this) {

        if(this.hasOwnProperty(index)) {

            var res = cb(this[index], index, this);

            if(res) {

                output.push(this[index]);

            }

        }

    }

    return output;

}

// Expected results:

const numbers = [1, 2, 3, 4];

console.log(numbers.myFilter(function (number) {

    return number % 2 === 0;

})); // Output: [2, 4]

console.log(numbers.myFilter(function (number, index) {

    return index % 2 === 0;

})); // Output: [1, 3]

console.log(numbers.myFilter(function (number, index, array) {

    return array.length % 2 === 0;

})); // Output: [1, 2, 3, 4]

**Some**: tạo phương thức mySome

Array.prototype.mySome = function(cb) {

    var output = false;

    for(var index in this) {

        if(this.hasOwnProperty(index)) {

            if(cb(this[index], index, this)) {

                output = true;

                break;

            }

        }

    }

    return output;

}

// Expected results:

const numbers = [1, 3, 3, 5];

console.log(numbers.mySome(function (number) {

    return number % 2 === 0;

})); // Output: false

console.log(numbers.mySome(function (number, index) {

    return index % 2 === 0;

})); // Output: true

console.log(numbers.mySome(function (number, index, array) {

    return array.length % 2 === 0;

})); // Output: true

**Every** : tạo phương thức myEvery

Array.prototype.myEvery = function(cb) {

    var output = true;

    for(var index in this) {

        if(this.hasOwnProperty(index)) {

            var res = cb(this[index], index, this);

            if(!res){

                output = false;

                break;

            }

        }

    }

    return output;

}

// Expected results

const numbers = [1, 3, 3, 5];

console.log(numbers.myEvery(function (number) {

    return number % 2 !== 0;

})); // Output: true

console.log(numbers.myEvery(function (number, index) {

    return index % 2 === 0;

})); // Output: false

console.log(numbers.myEvery(function (number, index, array) {

    return array.length % 2 === 0;

})); // Output: true

1. ***Function***

Tính chất của function: - Không thực thi khi định nghĩa

        - Sẽ thực thi khi được gọi

        - Có thể nhận tham số

        - Có thể trả về 1 giá trị

// Có thể truyền bất kì kiểu dữ liệu vào

 Các loại function: 1. Declaration function: gọi được trước khi định nghĩa

function declarationFunction() {

}

        2. Expression function: khởi tạo khi định nghĩa

var expressionFunction = function () {

}

        3. Arrow function

Cú pháp:

Function Name\_function(message){

// message: tham số

// đối tượng arguments: có thể truyền được nhiều biến

}

// showDialog('Test Message', 'Test2', 1234);

// Test Message: đối số

1. ***Object***

// Function: --> Phương thức / Method

// Others:   --> Thuộc tính  / Property

var SchoolKey = 'School';

var myInfor = {

    name: 'Vuong Duc Trong', // Property

    address: 'Nghe An, VN',

    age: 20,

    [SchoolKey]: 'PTIT', // thêm vào object

    getName: function() { // Method

        return this.name;

    }

};

// thêm key và value vào myInfor

myInfor['my-email'] = 'trongvuongduc24@gmail.com';

// lấy key ra

console.log(myInfor.address); // c1

console.log(myInfor['name']); // c2

delete myInfor.address;

console.log(myInfor);

***Object Constructor***

// Object constructor: xây dựng đối tượng 🡪 *Bản thiết kế của ngôi nhà*

// Gồm có: String, Number, Array, Date,...

function User(firstName, lastName, avatar){ // Mô tả thiết kế

    // mô tả thuộc tính, khi tạo từ đối tượng từ Object constructor User

    this.firstName = firstName;

    this.lastName = lastName;

    this.avatar = avatar;

    // Phương thức

    this.getName = function() {

        return `${this.firstName} ${this.lastName}`;

    }

}

// Khởi tạo đối tượng người dùng

var author = new User('Trọng', 'Vương', 'Avatar');

var user = new User('Linh Anh', 'Nguyễn', 'Avatar');

// Xem lại bản thiết kế có trùng với Object không

// console.log(author.constructor === User);

// thêm thuộc tính title = 'Boss'

author.title = 'Boss';

user.comment = 'Hello Anh';

// In ra Trọng Vương, Linh Anh Nguyễn

console.log(author.getName());

console.log(user.getName());

***Object Prototype***

// Object Prototype: nguyên mẫu tạo nên 1 đối tượng

// Nguyên liệu ngôi nhà

// thêm được 1 thuộc tính ở bên ngoài hàm User

User.prototype.className = 'CNTT';

// thêm phương thức vào hàm User

User.prototype.getClassName = function() {

    return this.className;

}

console.log(user1.className);

console.log(user2.getClassName());

// Những đối tượng được tạo ra từ User sẽ có những cái thuộc tính và phương thức

***Object Date***

var date = new Date();

var year = date.getFullYear();

var month = date.getMonth() + 1; // 0 - 11 --> 1 - 12

var day = date.getDate();

console.log(`${day}/${month}/${year}`);

// ***Ternary\_Operator***: toán tử ba ngôi

function run(fruits) {

    var result;

    switch(fruits) {

        case "Banana":

            result = "This is a Banana";

            break;

        case "Apple":

            result = "This is an Apple";

            break;

        default:

            result = "No fruits";

    }

    return result;

}

var course = {

    name: 'JavaScript',

    coin: 250

}

// if(course.coin > 0) {

//     console.log(`${course.coin} Coins`);

// }

// else {

//     console.log('Miễn phí');

// }

var res = course.coin > 0 ? `${course.coin} Coins` : 'Miễn phí';

console.log(res);

/\*

***Vòng lặp: Loop***

    1. for - Lặp với điều kiện đúng

    2. for/in - Lặp qua key của đối tượng

    3. for/of - Lặp qua value của đối tượng

    4. while - Lặp khi điều kiện đúng

    5. do/while - Lặp ít nhất 1 lần, sau đó lặp khi điều kiện đúng

\*/

// **FOR LOOP**

// for(var i = 1; i <= 1000; ++i){

//     console.log(`${i}`);

// }

// **FOR IN** :với key là chỉ số

// for(var key in myArray){

//     console.log(myArray[key]);

// }

// **FOR/OF LOOP**: in ra giá trị

var myString = 'JavaScript';

// for(var value of myString){

//     console.log(value);

// }

var myInfor = {

    name: 'Trong Vuong',

    age: 18,

    address: 'Nghe An, VN'

};

// chuyển myInfor về dạng mảng mới in ra được

// for(var value of Object.values(myInfor)){

//     console.log(value);

// }

// **Vòng lặp lồng nhau**

var myArray2 = [

    [1, 2],

    [3, 4],

    [5, 6],

];

for(var i = 0; i < myArray2.length; ++i){

    for(var j = 0; j < myArray2[i].length; ++j){

        console.log(myArray2[i][j]);

    }

}

// ***Include method***: chỉ có Array và String

var title = 'Responsive web design';

console.log(title.includes('web')); // tìm kiếm từ có trong xâu không

console.log(title.includes('web', 2)); // bắt đầu từ vị trí 2

var courses = ['JavaScript', 'PHP', 'Ruby'];

console.log(courses.includes('JavaScript'));

/\*

***Math Object:***

        Math.PI: trả về số pi

        Math.round(): làm tròn số

        Math.abs(): trị tuyệt đối

        Math.ceil(): làm tròn trên

        Math.floor(): làm tròn dưới

        Math.random(): chọn số thập phân ngẫu nhiên (0 <= x <= 1)

        Math.min(): giá trị nhỏ nhất

        Math.max(): giá trị lớn nhất

\*/

**// HTML DOM**: la quy chuan cua WAC dua ra

// DOM: Document Object Model : mo hinh doi tuong tai lieu

/\*

    1.Element: yeu to: html, head, body, h1, a, p, ...

    2.Attribute: thuoc tinh: class, id, title, href

    3.Test: chu

    DOM: get, change, add, delete HTML element

\*/

//-------------------------------------------------------

// Javascript: Browser | Server (NodeJS)

// Application Programing interface

// Browser: HTML -> DOM -> WEB API

/\*  **Element**:

        1.ID : getElementById() : tra ve 1 doi tuong

        2.class : getElementsByClassName() : tra ve nhieu doi tuong giong Array: []

        3.tag : getElementsByTagName() : tra ve nhieu doi tuong giong Array: []

        4.CSS selector : querySelector() : tra ve 1 doi tuong

                         querySelectorAll() : tra ve nhieu doi tuong

        5.HTML collection :

// innerHTML: lay duoc Element, Attribute, Text node: o ben trong

// outerHTML: tuong tu, lay duoc element ben ngoai, thay doi duoc element ben ngoai

\*/

/\*

**Attribute**: thuoc tinh

        class:

        id:

        title:

        href:

/\*

    cach 1 : truy xuat truc tiep qua thuoc tinh: phai hop le

// headingElement.id = 'heading';

    cach 2 : truy xuat phuong thuc ham: khong can phai hop le

// headingElement.setAttribute('class', 'heading');

\*/

\*/

**Text:**

// innerText: tra lai nhung gi ta nhin thay tren trinh duyet: Element node

// textContent: tra lai nhung gi trong textnode: ton tai Element node va text node

// geter: lay ra

// seter: dat lai

**DOM CSS:**

var divElement = document.querySelector('.red');

divElement.style.backgroundColor = '#f00';

document.querySelector('p').style.color = '#f05123';

// ClassList Property

// add : them class vao element

// contains : kiem tra class co nam trong element khong

// remove : xoa class trong element

// toggle : co thi bo khong co thi them

var boxElement = document.querySelector(".box");

console.log(boxElement.classList);

boxElement.classList.add('red', 'green', 'blue');

console.log(boxElement.classList.contains('red'));

boxElement.classList.remove('red');

boxElement.classList.toggle('red');

// **DOM events**: nguoi dung co nhung hanh vi tac dung len website: events co the lang nghe

// 1. Attribute events: viet thang cac the attribute vao html

// 2. Assign event using the element node: gan cac su kien qua Element node

EX:Cho trước thẻ button, các bạn hãy viết code JS sao cho khi click vào button sẽ đổi màu chữ button sang màu #fff.

var btn = document.querySelector('button');

btn.onclick = () => {

    btn.style.color = '#fff';

}

// 1. Input / Select

// var inputElement = document.querySelector('input[type="text"]');

// inputElement.oninput = function(e) {

//     console.log(e.target.value);

// };

// 2. Key up / Down

// var inputElement = document.querySelector('input[type="text"]');

// inputElement.onkeyup = function(e) {

//     console.log(e.which);

//     switch(e.which) {

//         case 27:

//             console.log('Exit');

//             break;

//     }

// };

**Method DOM Events:**

// 1**. preventDefault**: loai bo hanh vi cua trinh duyet tren 1 the html

var aElement = document.querySelectorAll('a');

// console.log(aElement);

for(var i = 0; i < aElement.length; ++i){

    aElement[i].onclick = function(e) {

        // console.log(e.target.href);

        if(!e.target.href.startsWith('https://f8.edu.vn')) {

            e.preventDefault(); // ngan chan hanh vi mac dinh cua trinh duyet

        }

    }

}

// 2**. stopPropagation**: su kien noi bot

// document.querySelector('div').onclick = function() {

//     console.log('DIV');

//     document.querySelector('button').onclick = function(e) {

//         e.stopPropagation();

//         console.log('Click me!');

//     }

// }

**// 1. Event listener**

// 1. Xử lý nhiều việc khi 1 event xảy ra

// 2. Lắng nghe / Hủy bỏ lắng nghe

// Event listener

// var btn = document.getElementById('btn');

// btn.addEventListener('click', function(e) {

//     console.log('Event 1');

// });

// btn.addEventListener('click', function(e) {

//     console.log('Event 2');

// });

// btn.addEventListener('click', function(e) {

//     console.log('Event 3');

// })

**// 2. JSON**

/\*

    JSON : // JavaScript Object Notation

        1. là 1 định dạng trao đổi dữ liệu nhẹ (chuỗi)

        2. không phụ thuộc vào ngôn ngữ \*

        3. "tự mô tả" và dễ hiểu

        Json: Number, Boolean, Null, Array, String, Object

        Mã hóa / giải mã

        Encode / Decode

        + Stringify: Từ JavaScript types -> JSON

        + Parse: Từ Json -> JavaScript types

\*/

var json = '["JavaScript", "PHP"]';

// console.log(JSON.parse(json));

// console.log(JSON.stringify(json));

**// 3. Promise**

**// Promise: xử lý thao tác bất đồng bộ**

/\*

**Sync / Async**

    đồng bộ / bất đồng bộ

    Bất đồng bộ:

    // setTimeout, setInterval, fetch,

    // XMLHttpRequest, file reading,

    // request animation frame

    // Callback

\*/

// sleep

// setTimeout(function() {

//     console.log(1);

// }, 1000);

// console.log(2);

// Pyramid of doom

// Callback hell <pain>

setTimeout(function() {

    console.log(1); // viec 1

    setTimeout(function() {

        console.log(2); // viec 2

        setTimeout(function() {

            console.log(3); // viec 3

            setTimeout(function() {

                console.log(4); // viec 4

            }, 1000);

        }, 1000);

    }, 1000);

}, 1000);

/ Promise: **pain**

// 1. new Promise

// 2. Executor

// 3 trạng thái:

    // 1. pendding

    // 2. fulfilled

    // 3. rejected

// Memory leak

var promise = new Promise(

    // Executor

    function(resolve, reject) {

        // logic

        // resolve(); // successful

        // reject(); // error

        // Fake call API

        reject('Co loi!');

    }

);

promise

    .then(function() {

        // successful status

        console.log('Successful!');

    })

        .catch(function(error) {

        // error status

        console.log(error);

    })

    .finally(function() {

        // done status

        console.log('Done');

    })

**// Chain**

function sleep(ms) {

    return new Promise(function(resolve) {

        setTimeout(resolve, ms);

    });

}

// tạo khoảng trễ mỗi lần in ra

sleep(1000)

    .then(function() {

        console.log(1);

        return sleep(1000);

    })

    .then(function() {

        console.log(2);

        return sleep(1000);

    })

    .then(function() {

        console.log(3);

        return sleep(1000);

    })

    .then(function() {

        console.log(4);

        return sleep(1000);

    })

**// Promise methods: resolve, reject, all**

// 1. Promise.resolve : chỉ đúng

// 2. Promise.reject : chỉ sai

// 3. Promise.all : chạy song song khi không phụ thuộc nhau

    // nếu 1 trong 2... có lỗi sẽ ko chạy nữa chuyển sang catch

// Thư viện: output luôn luôn là 1 promise

**// 1. Promise.resolve**

// var promise = Promise.resolve('Success!');

**// 2. Promise.reject**

var promise = Promise.reject('Error!');

promise

    .then(function(result) {

        console.log('result:', result);

    })

    .catch(function(err) {

        console.log('result:', err);

    })

**// 3. Promise.all**

var promise1 = new Promise(function(resolve) {

    setTimeout(function() {

        resolve([1]);

    }, 1000);

})

var promise2 = new Promise(function(resolve) {

    setTimeout(function() {

        resolve([2, 3]);

    }, 2000);

})

Promise.all([promise1, promise2])

    .then(function(result) {

        var result1 = result[0];

        var result2 = result[1];

        console.log(result1.concat(result2)); // nối mảng

    })

**// 4. Fetch**

// 1. Front-End: xây dựng giao diện người dùng

// 2. Back-End: xây dựng logic xử lý : Database

// API (URL): Application programing interface

// Cổng giao tiếp giữa các phần mềm

**// Backend (OK) => API (URL) => Fetch => JSON / XML**

**// => JSON.prase => JavaScript types**

**// => Render ra giao diện với HTML**

var postApi = 'https://jsonplaceholder.typicode.com/posts';

// Stream

fetch(postApi)

    .then(function(response) {

        return response.json();

        // JSON.parse: JSON -> JavaScript types

    })

    .then(function(posts) {

        var htmls = posts.map(function(post) {

            return `<li>

                <h2>${post.title}</h2>

                <p>${post.body}</p>

            </li>`;

        });

        var html = htmls.join('');

        document.getElementById('post-block').innerHTML = html;

    })

    .catch(function(err) {

        console.log('Error!');

    });

// CRUD

/\*

    Create: tạo mới => POST

    Read: lấy dữ liệu => GET

    Update: chỉnh sửa => PUT / PATCH

    Delete: xóa => DELETE

\*/

// Sử dụng Postman làm việc với REST API

**// 5. DOM location**

**// 6. Local storage**

**// 7. Session storage**

**// 8. Coding convention**

**// 9. Best Practices**

**// 10. Mistakes**

**// 11. Perfromance**