**Javascript cơ bản**

Javascript là gì ?

là một ngôn ngữ thông dịch : dịch ra mã máy từng dòng và nó thực hiên từng dòng . Hỗ trợ OPP, đa nền tảng nhỏ nhẹ.

-Chia ra làm phía client và phía sever :

+ client : là javascript thuần sử dụng trong những trình duyệt.

+ phía sever: học backend có thể học thêm notejs.

-Thêm, sửa, xóa trong html.

-React, Angular, Vue dựa trên 100% Javascipt.

-Ngôn ngữ biên dịch như là c ,c++ ,..: biên dịch 1 lần đoạn code của mình 1 lần ra mã máy rồi nó mới chạy.

Các phiên bản Javarscript hiện nay:

..-> ES5 -> ES6/ES2015 -> ES7/ES2016 -> ES8/ES2017

1. **BIẾN VÀ KIỂU DỮ LIÊU**

Biến lưu trữ thông tin mà người lập trình muốn.

\*Khác với C, C++, C#, Javar: -> phân ra nhiều loại int, float ,double…

- Có 5 kiểm dữ liệu **nguyên thủy( cơ bản** ):

1. Number : Số thực dấu chấm động, số thập phân, sô nguyên.
2. String: Một dãy các kí tự, được dùng cho đoạn text.
3. Boolean: Kiểu dữ liệu luận ly. Có thể true hoặc false.
4. Underfined: Kiểu dữ liệu cho một biến chưa được gán giá trị.
5. Null: Có thể hiểu là không tồn tại, phải được gán có chủ đích.

* Lưu ý: Javascript có kiểu dữ liệu động ( dynamic typing ):

Các kiểu dữ liệu được tự động tiên đoán cho một biến khi gán giá trị.

- **Everything else**...: Array, Function, Object, Dates, Wrapper for Numbers, Strings, Booleans.

1. **ĐỘT BIẾN GIÁ TRỊ VÀ ÉP KIỂU**

-Ep một kiểu dữ liệu khác sang một kiểu dữ liệu tương đương.

-Đột biến giá trị ( đột biến biến): ví dụ: age:28 và chúng ta đột biến age:abc

.

1. **TOÁN TỬ GÁN**

|  |  |  |
| --- | --- | --- |
| Tên | Viết Tắt | Y Nghia |
| Toán tử gán | X = y | X = y |
| Cộng xong rồi gán | X +=y | X=x+y |
| Trừ xong rồi gán | X -=y | X=x-y |
| Nhân xong rồi gán | X \*=y | X=x\*y |
| Chia xong rồi gán | X /=y | X=x/y |
| Chia lấy dư xong rồi gán | X % y | X = x % y |
| Lũy thừa y xong rồi gán | X \* \*y | X =x\*\*y |
| Cộng một xong rồi gán | X++ | X =x+1 |
| Trừ một xong rồi gán | X-- | X =x-1 |

Toán tử typeof -> Dùng để xác định kiểu dữ liệu của biến tại thời điểm đó.

1. **TOÁN TỬ SO SÁNH**

Các toán hạng với nhau trả về giá trị logic ( boolean )

>, <, >=, <= , ==, === ,!= ,!==

-Grouping: gom nhóm độ ưu tiên. Bằng ngoặc.

1. **CẤU TRÚC ĐIỀU KIỆN VÀ RẼ NHÁNH**

-Toán tử logic (AND, OR, NOT):

+ AND (&&) => trả về true nếu tất cả điều kiện đều đúng.

+ OR ( || ) => trả về true chỉ cần thỏa mãn một trong số nhiều điều kiện.

+ NOT ( ! ) => phủ định true/ false value.

-Câu lệnh rẽ nhánh: if / else

-Câu lệnh rẽ nhánh: switch đầy đủ: break ngắt. so sánh này chỉ trả về 2 trường hợp hoặc là true hoặc là false , trả về luận lý;

1. **Truthy VÀ Falsy TRONG JAVARSCRIPT**

\*Tình huống gặp phải: var:

var varA; // undefiend là false.

//To do …

if( varA ) { //ép kiểu biến này về Boolean true/ false

//Condition varA ???

};

**Falsy:** Khi ép kiểu về Boolean, bản thân nó sẽ trả về giá trị là false khí dùng biểu thức điều kiện.

+ 6 giá trị thuộc falsy: Null, Undefiend, NaN, 0, ‘’, false. 6 giá trị này ép kiểu về Boolean thì sẽ trả về kết quả là false.

- NaN: một chuỗi nhưng không phải là số , ép kiểu về số nó sẽ ra NaN.

**Truthy:** Tất cả những giá trị khi chúng ta ép kiểu về Boolean, bản thân giá trị nó là true trong biểu thức điều kiện.

\* Nằm ngoài Falsy thì sẽ là true.

\* So sánh khác nhau giữa dấu ‘==’ và dấu ‘===’.

‘==’: nếu so sánh thì nó tự động ép kiểu về 1 giá trị tương đương, chỉ so sánh giá trị thôi.

‘===’: so sánh kiểu dữ liệu và giá trị thỏa mãn cả hai mới thực hiện. === nghiêm ngặt hơn ==.

1. **HÀM, THỦ TỤC ( FUNCTION, PROCEDURE )**

\_ Hàm: ( Chia nhỏ code của chúng ta ra phục vụ một chức năng nào đó, tránh thực hiện thao tác lặp đi lặp lại giống nhau. Được sủ dụng lại và gọi nhiều nới khác nhau nếu chúng ta cần sử dụng ).

+ là một đoạn chương trình độc lập, thực hiện một khối chức năng nào đó.

+ sau khi thực hiện xong => trả về một giá trị cho chương trình gọi nó.

* **HÀM:** sau khi sử lí xong sẽ trả về giá trị mới, lấy giá trị mới đó để thực hiện một thao tác tiếp theo . Trong hàm sẽ sử dụng từ khóa return.
* **THỦ TỤC:** Chỉ thực hiện thao tác không trả về giá trị.

**\*Các cách khai báo:**

**Debugger ; //** Kiểm tra debug lỗi.

**-Cách 1:**

fuction calAge( year ) {

Return …

}

**-Cách 2:**

Var calAge2 = function( year ) {

Return …

}

**-Cách 3:** => Arrow function , thường dùng trong một số framework: Reactjs

Var calAge3 = ( year ) ={

Return …

}

1. **MẢNG VÀ ĐỐI TƯỢNG ( ARRAY AND OBJECT )**

\*Tình huống : Một lớp học có 50 học sinh. Một hệ thống phải lưu trữ thông tin tên của 50 học sinh đó.

+ Không thể tạo 50 biến riêng biệt. => Giải pháp là tạo 1 mảng chứa thông tin 50 hs.

Mảng : sẽ có nhiều phần tử. Mỗi phần tử sẽ có 2 thông tin cần lưu ý đó là giá trị ( value ) và chỉ số ( index ).

Có 2 cách để tạo array:

Cach 1:

Var arrStudent = [ ‘nv a’ ,’nv b’, ‘nv c’, ‘nv d’ ];

+length: thuộc tính tổng của mảng.

+arrStudent[chỉ số]: truy xuất phần tử trong mảng.

Cach 2:

Var arrStudent = new Array( ‘nv a’ , ‘nv b’, ’nv c’, ’nv d’ );

* **Lưu ý:**

-C, C++ ,C# , Javar => array chỉ lưu chữ được 1 kiểu dữ liệu.

-Trong Javascript bản chất của array là một Object. Bản chất của array là được hiện thực từ 1 Object , trong array chứa nhiều kiểu khác nhau.( có thể array lồng array ).

1. **CÁC PHƯƠNG THỨC HỖ TRỢ CHO ARRAY ( ARRAY METHOR )**

-Phương thức thêm 1 phần tử vào phía sau của mảng push: arr.push( giá trị ). Có thể thêm nhiều phần tử.

-Phương thức thêm 1 phần tử vòa phía đầu của mảng arr.unshift( giá trị ). Có thể thêm nhiều phần tử.

-Phương thức remove xóa đi phần tử cuối cùng của mảng arr.pop().

-Phương thức remove xóa đi phần tử đầu tiên của mảng arr.shift().

-Phương thức indexOf kiểm tra xem phần tử mình cần tìm kiếm có trong mảng hay không, return giá trị đầu tiên xuất hiện tron mảng arr.indexOf(giá trị) // trả về vị trí của phần tử xuất hiện đầu tiên. Nếu nó trả về -1 thì phần tử đấy không tồn tại trong mảng.

-Phương thức lastIndexOf return về phần tử xuất hiện cuối cùng trong mảng. Ngược lại với indexOf.

🡺Nếu như indexOf và lastIndexOf trả về cùng 1 giá trị giống nhau => Giá trị đó tồn tại trong mảng.

Truy xuất mảng tới vị trí không hợp lệ => undefined.

Gán giá trị cho vị trí không hợp lệ cũng sinh ra phần tử trong mảng vào cuối mảng, mình có thể thêm trực tiếp bằng cách 2 này: arr[ arr.length ]=’test’;

1. **ĐỐI TƯỢNG ( Object ) TRONG JAVASCRIPT( Đối tượng trong javascript )**

-Gần tương tự như array.

\* Tình huống có 1 lớp học có 50 học sinh. Mỗi học sinh cần phải lưu ( Họ tên, Ngày tháng năm sinh, Điểm toán , Lý, Hóa, Điểm trung bình) của 50 học sinh.

\_Trong tình huống huống trên, một học sinh là một đối tượng (object):

+ Object này có những thuộc tính ( properties ) như là họ tên, năm sinh, điểm thi.

+ Object học sinh có phương thức ( method ) là tính điểm trung bình, phụ thuộc vào những giá trị được lưu trong properties điểm toán, lý, hóa.

* Method: có thể hiểu là function gắn liền với tưng Object.

Một Object có nhiều ( properties ) và ( method ).

2 Cách khai báo mảng.

-Cách 1: ví dụ: tạo một đối tượng.

Lưu thuộc tính dưới dạng key : value

Var student = {

fullName : ‘Nguyễn Văn A’,

year : 1990,

scores { Object lồng nhau

maths: 9,

physics: 7.5,

chemistry: 5,

}

Xem method như một function

calAverage: function( ){

Tính toán dựa trên scores

Var maths = this.scores.maths;

Var physics = this.scores.physics;

Var chemistry = this.scores.chemistry;

Return (maths + physics + chemistry)/3;

}

}

+Truy xuất : tới thuộc tính tên student.fullName.

+Truy xuất : tới phương thức student.calAverage( );

+Truy xuất : có thể truy xuất giống như mảng student[‘demo-name], đối với thuộc tính mà đặt tên kiểu như này demo-name.

+John.calBMI() -> lời gọi hàm.

+BMI: calBMI -> nó sẽ xem BMI như một cái biến trỏ địa chỉ đến hàm. Trỏ địa chỉ hàm được lưu trong bộ nhớ của máy tính.

Gán : tương tự như array. Object cũng gán một phần tử không hợp lệ student.lastName = ‘ giá trị ’.

\* Thao tác với Object sẽ có một từ khóa gọi là this => Con trỏ this => Trỏ tới đối tượng hiên tại.

-Cách 2:

Var student2 = new Object();

Student2.fullName(‘ Nguyễn Văn B ’);

Student2.scores{

Maths: 8,

Physics: 7,

Chemistry: 5,

}

Có thể copy phương thức của cách 1:

Student2.calAverage = Student1.calAverage con trỏ this tự trỏ vào student2.

Có thể viết phương thức tương tự như cách 1.

1. **CẤU TRÚC LẶP ( For , While Loop )**

Vòng lặp: dùng để thực thi một đoạn code nào đó nhiều lần liên tiếp.

*Các dạng vòng lặp trong JavaScript:*

for, for/in , while , do / while;

* *For loop:*

For ( statement 1 ; statement 2 ; statement 3)

{ //code block to be executed }

* Statement 1: Thực thi một lần trước khi thực hiện đoạn code bên trong ( code block ).
* Statement 2: Định nghĩa điều kiện để có thể chạy vòng lặp tiếp theo. ( tránh vòng lặp vô tận ).
* Statement 3: Chạy xong đoạn code ở giữa đều thực thi lại câu lệnh thứ 3.
* *For / in :*

Thường được lặp trong 1 Object: Không phải là một mảng nên nó không có thuộc tính length.

For( key in student ){

Console.log(“ key = ”, key, “ | value = ”, student[key]);

}

* *While*

Cú pháp ( syntax ):

While ( condition ){

//code block to be excuted

}

+ While nếu như điều kiện đúng thì nó sẽ lặp, sai thì thôi.

* *Do / While*

Cú pháp ( syntax ):

Do{

//code block to de excuted

}

While ( condition );

+ Do / While làm trước rồi mới lặp sau.

1. **Bind, Call and Apply**

The powerful call, apply and bind methods and in a nutshell( tóm lại ) , these methods allow( cho phép ) to call a function and set this variable manually ( thủ công ).

1. **asdasds**