**JavaScript**

1. Javascript Nền Tảng
   * 1. Lịch sử phát triển của JavaScript
     2. JavaScript khác với Java và JScript như thế nào?
     3. JavaScript nó có thể làm đựợc gì?Ứng dụng của JavaScript như thế nào?
     4. Các cú pháp của JavaScript
        1. Chú Thích
        2. Khai báo Biến
        3. Đối tượng
        4. Cấu trúc điều khiển
        5. Toán tử điều kiện
        6. Vòng lặp while
        7. Vòng lặp do.. while
        8. Vòng lặp for
        9. Vòng lặp for .. in
        10. switch
2. Lập trình hướng đối tượng trong Javascript
   1. Lập trình OOP trong Javascript.

**Lịch sử phát triển của JavaScript**

-JavaScript theo phiên bản hiện nay mà chúng ta đang sử dụng,là một Ngôn ngữ lập trình kịch bản dựa trên đối tựơng.JavaScript đựoc sử dụng rộng rãi cho các trang web hiện nay,nhưng nó cũng đưọc dùng để tạo khả năng viết script bằng việc sử dụng các đối tượng có sẳn trong các ứng dụng.-Vậy JavaScript xuất hiện từ khi nao?Nó vốn đưọc phát triển bởi Brendan Eich tại hãng truyền thông NetScape với tên đầu tiên là MoCha, qua một thời gian tiếp theo thì Javascript đổi tên là LiveScript.Và cái tên Javascript là cái tên hiện tại ngày nay.JavaScript có cấu trúc tương tự như C. Các tập tin của Javascript được lưu với định dạng là .js(vd: demo.js).-Phiên bản mới nhất của JavaScript là phiên bản 1.5, tương ứng với ECMA-262 bản 3. ECMAScript là phiên bản chuẩn hóa của JavaScript. Trình duyệt Mozilla phiên bản 1.8 beta 1 có hỗ trợ không đầy đủ cho E4X - phần mở rộng cho JavaScript hỗ trợ làm việc với XML, được chuẩn hóa trong ECMA-357.

**JavaScript khác với Java và JScript như thế nào?**

-Cùng với sự ra đời của Java như một hiện tượng thì LiveScript cũng đã đổi thành JavaScript để thu hút những Người Lập Trình hơn.Suy cho cùng thì Java và JavaScript hoàn toàn khác nhau(Bạn đừng nhầm lẫn Chúng giống nhau là java mà cho rằng chúng có họ hàng),ngoại trừ cú pháp của chúng giống với C-Sau thành công của JavaScript thì Microsoft bắt đầu phát triển JScript, một ngôn ngữ có cùng ứng dụng và tương thích được với JavaScript.JScript được bổ sung vào IE 3.0

**JavaScript nó có thể làm đựợc gì?Ứng dụng của JavaScript như thế nào?**

-Javascript có thể làm được rất nhiều thứ chẳng hạn:

Nó có thể tạo ra một HTML động

Nó có thể thiết kế một giao diện

Nó có thể tạo một sự kiện (Event) cho các button.

Nó cũng có thể tạo ra một Cookies.

Và còn nhiều thứ khác nữa mà bạn khó có thể tưởng tượng được lợi ích của nó.Ta có thể tìm hiểu kỹ hơn.

-Ứng dụng của Javascript.

Hiện nay,có rất nhiều trang web trên mạng sử dụng JavaScript để thiết kế trang web động và một số hiệu ứng hình ảnh thông qua DOM để tạo sức thu hút ngững con mạng.  
-JavaScript dùng để xử lý một số thao tác không thể thực hiện được với một trang HTML bình thường như kiểm tra thông tin nhập vào,username,pass,hay tự động thay đổi hình ảnh...vv -Một số công nghệ nổi bậc kết hợp JavaScript tương tác với DOM như:DHTML, Ajax và SPA.

**JavaScript: JavaScript Standard Objects,Browser Object Model, Document Object Model**

Có 3 loại đối tượng bạn làm việc với khi sử dụng JavaScript cho website:

**JavaScript Standard Objects** và đối tượng người dùng quy định.

function f {return 2;}

Function

Array

Date

Regex

JSON

Math

**Browser Object Model** : Đây là đối tượng window trong trình duyệt

đối tượng window của trình duyệt là đối tượng toàn cầu. Khi bạn định nghĩa một biến toàn cầu trong JavaScript trong trình duyệt, nó là một thuộc tính của đối tượng window.

VD: alert("3") là giống như window.alert("3")

window.alert("hi")

window.open("http://example.com/")

window.setTimeout(…)

window.setInterval(…)

window.encodeURI(…)

window.innerWidth(…)

window.localStorage(…)

window.history(…)

window.location(…)

**Document Object Model** :Thao tác các phần tử HTML và styles . hệ thống đối tượng này được gọi là DOM (Document Object Model).

Insert/remove HTML element.

Change element's attributes. (such as class, id, name, width, etc.)

Change element's content.

Change element's CSS style.

**Các cú pháp của JavaScript**

**Chú Thích:**

-Chú Thích dòng: //chú thích ở đây  
-Chú Thích Khối: /\* chú thích nhiều dòng\*/

**Khai báo Biến**

-Khi đặc tên cho một biến ta chú ý không được bắt đầu bằng một số (vd:1abc), hay chúng ta không được dùng các ký tự đặt biệt (vd:\*,+,-...)

var X; ->[hợp lệ]

var x; ->[hợp lệ]

var \_x; ->[hợp lệ]

var 1x; ->[không hợp lệ]

- **Cách khai báo biến**

Chúng ta sử dụng từ khóa var để khai báo một biến,các biến có thể nắm giữ giá trị hoặc là không

var x; ->không nắm giữ giá trị

var x=5; ->nắm giữ giá trị là 5

- **Phạm vi của biến**

Khi một biến khai báo có thể là Local hoặc là Global  
-Một biến gọi là Local khi chúng được khai báo trong một function(hàm) để phục vụ cho hàm đó.  
-Một biến gọi là Glocal khi chúng được khai báo nằm bên ngoài các function(hàm) để phục vụ cho các hàm

<script language="JavaScript">

var a; //bien Glocal

var b=2; //bien Glocal

var result=0; //bien Glocal

function myFunction1(){

var b=10; // bien local

result=a+b;

document.write("Kat Qua cua ham myFunction1 la : "+result+"<br>");

}

//////

function myFunction2(){

result=a+b;

document.write("Kat Qua cua ham myFunction2 la : "+result);

}

///

function calculate(){

var inputText\_a=document.getElementById("so\_a");

a=parseInt(inputText\_a.value);

myFunction1();

myFunction2();

}

function init(){

var btSum\_a\_b=document.getElementById("btSum\_a\_b");

btSum\_a\_b.onclick=calculate;

}

window.onload=init;

</script>

**Đối tượng**

Trong javaScript có 2 kiểu dữ liệu:Kiểu Cơ bản và Kiểu Đối Tượng

- **Kiểu Cơ Bản**:là kiểu mà chỉ có một giá trị duy nhất.Sau đây là Bảng Kiểu Cơ bản(Data types)

Number: kiểu dữ liệu thuộc về số (4, 5.3, hoặc là 789)

String: kiểu dữ liệu thuộc về chuổi ("Hello to you!","554-212-023","KJH566XHJD")

Boolean: Có 2 giá trị True hoặc False

var str="Hello!" ->biến str là kiểu String của javascript

- **Kiểu Đối Tượng** là một thực thể có tên xác định và có thuộc tính(attribute) trỏ đến giá trị,hàm hoặc là một đối tượng khác.(Đối tượng có thể là do javascript cung cấp hoặc là chính do chúng ta tạo ra)

var checkbox=document.getElementById("ch1"); //lấy một đối tượng Checkbox

var result=checkbox.checked; //lúc này result=true hoặc false

javaScript cung cấp cho ta một số đối tượng sau:

Anchor, Applet ,Area , Array, Boolean, Button, Checkbox, Date, Document, Event,

FileUpload, Form, Frame, Function, Hidden, History, Image, Layer, Math, Object,

Reset, Screen, String, Submit, Text, Textarea, Window, Link, Location, Navigator,

Number, Option, Password, Radio, RegExp, Select

**Cấu trúc điều khiển**

<script type="text/javascript">

var d = new Date()

var time = d.getHours()

if (time<10)

{

document.write("<b>Good morning</b>");

}

else if (time>10 && time<16)

{

document.write("<b>Good day</b>");

}

else

{

document.write("<b>Hello World!</b>");

}

</script>

**Toán tử điều kiện**

var x=(ham>=1)?1:0; //nếu giá trị ham lớn hơn hoặc bằng 1->x=1;ngược lại x=0

**Vòng lặp while**

Vòng lặp while có mục đích lặp đi lặp lại một khối lệnh nhất định cho đến khi biểu thức điều kiện trả về false. Khi dùng vòng lặp while phải chú ý tạo lối thoát cho vòng lặp (làm cho biểu thức điều kiện có giá trị false), nếu không đoạn mã nguồn sẽ rơi vào vòng lặp vô hạn, là một lỗi lập trình. Vòng lặp while thường được dùng khi chúng ta không biết chính xác cần lặp bao nhiêu lần.

var array=new array[1,"asd","fdg",3];

while () {

khối lệnh cần thực hiện nếu biểu\_thức\_điều\_kiện trả về true;

}

**Vòng lặp do.. while**

<script language="JavaScript">

while (0 > 1)

{

document.write(" 0 thì bé hơn 1"); // Câu lệnh này sẽ không bao giờ được thực hiện

}

do

{

document.write("Hello"); // ta thấy được dòng Hello một lần duy nhất cho dù điều kiện là sai

} while (0 > 1);

</script>

Tương tự như vòng lặp while.Nhưng có điểm khác nhau là :Đối với while nếu gặp giá trị biều thức là false lần đầu tiên chạy vòng lặp thì khối lệnh sẽ không bao giời được thực thi.Còn đối với do..while thì ít nhất thực hiện một lần cho dù điều kiện sai.

**Vòng lặp for**

Chúng ta thường dùng khi muốn thực hiện một khối lệnh nào đó mà chúng ta biết được số lần đó.

<script language="javascript">

for(var i=1; i<10; i++){

document.write("Hello "+i); //thực hiện cho đến khi i=9

}

</script>

**Vòng lặp for .. in**

Dùng để duyệt qua tất cả các phần tử trong mảng hay Chúng ta cũng có thể duyệt qua tất cả các thuộc tính của một Object.

<script language="javascript">

function Sinhvien(){

this.ten="A";

this.tuoi=20;

}

var sv=new Sinhvien;

for(var attr in sv){

document.write(attr+"<br>" );

}

</script>

**switch**

Cấu trúc switch cũng tương tự như if..else nhưng khác ở chổ nó sẽ kiểm tra bằng với kết quả được đưa ra.Nếu trường hợp mà biến không trùng với những kết quả cần kiểm tra thì nó sẽ thực thi lệnh default

<script language="javascript">

var bien="Cat";

switch(bien){

case "Cat":document.write("Day la Cat");break;

case "Dog":document.write("Day la khong Cat");break;

default:document.write("Khong la gi ca");

}

</script>

**Lập trình hướng đối tượng trong Javascript**

**Tính đóng gói (encapsulation)**:cung cấp những phương thức gọi đến các đối tượng trong javaScript như là một Class:

<script language="JavaScript">

obj =new Object; //tạo ra đối tượng Object

obj.name="ABC"; //thêm thuộc tính name cho object

obj.address="q1 tphcm"; //thêm một thuộc tính nữa

document.write("kqua : "+obj.name); //hiển thị name của object

</script>

-Chúng ta có thể tạo ra vô số các thuộc tính cho đối tượng Object tại mọi lúc ta cần.Như vậy lúc này obj của chúng ta có 2 thuộc tính name và address ngoài ra nó còn có thêm thuộc tính là constructor có giá trị là Object do môt hàm function trong javascript thông qua thuộc tính prototype (Object.prototype).Chúng ta sẽ giải thích prototype sau.

Tạo ra một Class và một constructor cho đối tượng.

Để tạo ra một class bằng cách dùng một hàm đơn giản.Khi mà hàm được gọi với phép toán new thì Server sẽ tạo ra một class.sau đó javascript sẽ tạo ra một đối tượng và khi đó đối tượng sẽ gọi hàm constructor.Trong constructor biến this trỏ đến object vừa được tạo ra

function Foo(){

this.x = 1;

this.y = 2;

}

obj = new Foo;

Trong javascript có thể kế thừa các trường từ những đối tượng khác, được gọi là prototype.Khi giá trị được trả về cho một biến thì đầu tiên javascript kiểm tra xem biến đó đã được tạo trong object chưa,nếu chưa thì nó tiếp tục tìm kiếm trong prototype của object vừa được tạo ,sau đó thay đổi giá trị của prototype.

Ex: Nếu ta muốn tạo ra đối tượng x từ hàm constructor B.

<script language="JavaScript">

Object.prototype.inObj = 1;

function A(){

this.inA = 2;

}

A.prototype.inAProto = 3; //them mot truong cho A thong qua prototype

B.prototype = new A; // B kế thừa A

B.prototype.constructor = B;

function B(){

this.inB = 4;}

B.prototype.inBProto = 5;

x = new B;

document.write(x.inObj + ', ' + x.inA + ', ' + x.inAProto + ', ' + x.inB + ', ' + x.inBProto);

Ket qua in ra man hinh la bao nhieu?

</script>

Định nghĩa một phương thức và gọi phương thức của một Class.

<script language="JavaScript">

Object.prototype.inObj = 1;

function A(){

this.inA = 2;

this.addA=function (){//định nghĩa hàm trong constructor(this tham chiếu đến A)

var x="Hello";

return x;

}

}

a=new A;

document.write("Kqua : "+a.addA());//gọi phương thức addA của A

Ket qua in ra man hinh la gi?

</script>

Ngoài ra chúng ta còn có thể định nghĩa một hàm trong prototype của constructor :

<html>

<head>

<script language="JavaScript">

function Foo(){

this.x = 1;

}

Foo.prototype.AddX = function(y){ //định nghĩa ra một phương thức

return this.x+=y;

}

foo=new Foo;

var re=foo.AddX(5);

document.write(re);

</script>

</head>

<body>

</body>

</html>

**Tính đa hình (polymorphism):** Thể hiện thông qua việc gửi các thông điệp (message). Việc gửi các thông điệp này có thể so sánh như việc gọi các hàm bên trong của một đối tượng. Các phương thức dùng trả lời cho một thông điệp sẽ tùy theo đối tượng mà thông điệp đó được gửi tới sẽ có phản ứng khác nhau. Người lập trình có thể định nghĩa một đặc tính (chẳng hạn thông qua tên của các phương thức) cho một loạt các đối tượng gần nhau nhưng khi thi hành thì dùng cùng một tên gọi mà sự thi hành của mỗi đối tượng sẽ tự động xảy ra tương ứng theo đặc tính của từng đối tượng mà không bị nhầm lẫn.

Các đối tượng khác nhau cũng có thể có các phương thức cùng tên.Vì thế khi gọi hàm ta nên gọi chính xát tên

function Person(age, weight) {

this.age=age;

this.weight=weight;

this.getInfo=function() {

return "I am " + this.age + " years old " +

"and weighs " + this.weight +" kilo.";

}

}

function Employee(age, weight, salary){

this.salary=salary;

this.age=age;

this.weight=weight;

this.getInfo=function() {

return "I am " + this.age + " years old " +

"and weighs " + this.weight +" kilo " +

"and earns " + this.salary + " dollar.";

}

}

Employee.prototype= new Person();

Employee.prototype.constructor=Employee;

// The argument, 'obj', can be of any kind

// which method, getInfo(), to be executed depend on the object

// that 'obj' refer to.

function showInfo(obj) {

document.write(obj.getInfo()+"<br>");

}

var person = new Person(50,90);

var employee = new Employee(43,80,50000);

showInfo(person);

showInfo(employee);

**Tính kế thừa (inheritance):** Đặc tính này cho phép một đối tượng có thể có sẵn các đặc tính mà đối tượng khác đã có thông qua kế thừa. Điều này cho phép các đối tượng chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại. Tuy nhiên, không phải ngôn ngữ định hướng đối tượng nào cũng có tính chất này.

Sử dụng từ khóa prototype để kế thừa một Super-Class(Lớp cha).Lớp con có thể kế thừa các phương thức cũng như các trường của lớp cha mà chúng không cần khai báo.ngoài ra lớp con còn có thể có thêm các trương khác nữa

function A() { // Định nghĩa lớp Cha

this.x = 1;

}

A.prototype.DoIt = function() // Viết Method cho lớp cha

{ this.x += 1;}

B.prototype = new A; // Định nghĩa sub-class

B.prototype.constructor = B;

function B(){

A.call(this); // gọi constructor của lớp cha

this.y = 2;}

B.prototype.DoIt = function() { // Định nghĩa Method cho lớp con B

A.prototype.DoIt.call(this); // gọi một phương thức của lớp cha

this.y += 1;}

b = new B;

document.write((b instanceof A) + ', ' + (b instanceof B) + '<BR/>');

b.DoIt();

document.write(b.x + ', ' + b.y);