It.Kma

Tìm hiểu Javascript

Khoa CNTT – HvKTMM
Giảng viên: ThS. Lê Đức Thuận

Javascript

1. Giới thiệu Javascript
2. Cách hiển thị trong JS
3. Câu lệnh trong Javascript

1. Giới thiệu Javascript

- Javascript là một ngôn ngữ lập trình kịch bản dựa vào các đối tượng có sẵn hoặc do người dùng tự định nghĩa.
- Javascript được sử dụng phổ biến được hỗ trợ bởi hầu hết tất cả các trình duyệt.
- Hiện nay được sự hỗ trợ của nhiều ông lớn như: google, facebook, …
  - Angular Js 1-2
  - Reactjs
  - Jquery
  - Nodejs
  - Vuejs
  - Knockoutjs
  - ….

1. Giới thiệu Javascript

Sử dụng JS

- Javascript được viết ở bên trong thẻ head

```
<head>
    <title>Bắt đầu học javascript</title>
    <meta charset="utf-8">
    <script type="text/javascript">
        function myFunction() {
        document.getElementById("demo").innerHTML =
            "Chế độ javascript được kích hoạt";
    }
    </script>
</head>
```

- Hoặc được viết trong thẻ <body> đặt ở bất kỳ đâu.

```
<body>
    <h1>Bắt đầu Javascript</h1>
    <p id="demo">Demo</p>
    <button type="button" onclick="myFunction()">Kiểm tra javascript</button>
    <script type="text/javascript">
        function myFunction() {
        document.getElementById("demo").innerHTML = "Chế độ javascript được kích hoạt";
    }
    </script>
</body>
```

- Javascript được viết ở file bên ngoài

```
<script type="text/javascript" src="../JS/file_JS.js"></script>
```

2.Cách hiển thị dữ liệu ra màn hình

- Cách 1: Sử dụng alert(): Hiển thị trong một hộp thoại thông báo.
  - Cú pháp: alert("nội dung hiển thị");
- Cách 2: document.write(): nội dung hiển thị ngay tại vị trí mà nó được đặt trong trang web.
  - Cú pháp: document.write("nội dung muốn hiển thị")
  - Nếu sử dụng trong thuộc tính onclick của button thì tất cả nội dung hiển thị trong website sẽ bị thay thế bằng nội dụng trong JS khi click vào button.
- **Cách 3: console.log**("nội dung"); → In nội dung ra màn hình console

2.Cách hiển thị dữ liệu ra màn hình (tt)

- Cách 4: Sử dụng hiển thị trong các thẻ html:
  - Theo id
  - Theo class
  - Tên thẻ

Ví dụ: document.getElementById(“ID của thẻ”).innerHTML: Hiển thị bên trong một thành phần xác định.

- Cú pháp:

```
document.getElementById("id của phần tử").innerHTML
= nội dung muốn hiển thị;
```

2.Cách hiển thị dữ liệu ra màn hình (tt)

- Chú ý:
  - Nội dung hiển thị là một chuỗi ký tự, chuỗi đó phải nằm trong dấu nháy kép hoặc nháy đơn.
  - Nếu nội dung là số thì có thể (hoặc không cần) đặt trong nháy kép hoặc đơn.
  - Chuỗi trong nháy kép (đơn) thì không được phép chứa ký tự nháy kép (đơn) bên trong.
  - Bên trong chuỗi có thể dùng thẻ HTML.

3. Câu lệnh trong JS

- Câu lệnh được kết thúc bằng dấu ";"
- Chương trình JS gồm nhiều câu lệnh, thực thi lần lượt từ trên xuống dưới
  - Nếu một lệnh bị lỗi thì tất cả các lênh ở phía dưới không được thực thi.
  - Có thể có nhiều thẻ <script> bên trong HTML => bản chất vẫn giống 1 script được thực hiện từ đầu đến cuối.
  - Nên để JS ở phía cuối của thẻ body (khi đó các thẻ HTML đã có và thực thi)
  - Comment trong JS
    - Inline: //
    - Block: /* */

3. Câu lệnh trong JS
Biến

- Khai báo biến:
  - Var tên biến;
  - Ví dụ: var hoten; var x, y, z;
- Gán giá trị cho biến:
  - Tên biến = giá trị (giá trị nằm trong dấu nháy kép hoặc đơn nếu là ký tự)
  - Ví dụ: x = 10; hoten = “Lê Lung Linh”;
- Gán giá trị khi khai báo và gán đồng thời
  - Var x = 10, y, z = “Hà Nội”;
- Tên biến: chữ, số, gạch dưới (_), không bắt đầu bằng số.
- Biến chưa gán giá trị thì nó nhận giá trị mặc định là undefined.

3. Câu lệnh trong JS
Toán tử

- Các toán tử thông dụng: + , - , * , / , %
- Quy tắc sử dụng toán tử
  - Số + số = số
  - Số + chuỗi = chuỗi
  - Chuỗi + chuỗi = chuỗi
- Toán tử ++, --

```
<script>
    //toán tử tăng và giảm đi 1
    var a = 10;
    var b = 10;
    var c = 10;
    var d = 10;
    document.write("<br><br>giá trị của a = " + ++a);
    document.write("<br>Giá trị mới của a = " + a);
    document.write("<br>giá trị của b = " + --b);
    document.write("<br>Giá trị mới của b = " + b);
    document.write("<br>giá trị của c = " + c++);
    document.write("<br>Giá trị mới của c = " + c);
    document.write("<br>giá trị của d = " + d--);
    document.write("<br>Giá trị mới của d = " + d);
</script>
```

giá trị của a = 11
Giá trị mới của a = 11
giá trị của b = 9
Giá trị mới của b = 9
giá trị của c = 10
Giá trị mới của c = 11
giá trị của d = 10
Giá trị mới của d = 9

3. Câu lệnh trong JS
Toán tử gán

Toán tử	Ví dụ	mô tả
=	X = y	
+=	X += y	X = x+y
-=	X -= y	X= x – y
*=	X *= y	X = x * y
/=	X /= y	X = x / y
%=	X %= y	X = x % y

3. Câu lệnh trong JS
Toán tử so sánh và logic

Toán tử	Ý nghĩa
>	
>=	
<	
<=	
==	Bằng giá trị (không phân biệt kiểu dữ liệu
===	Bằng giá trị (phải cùng kiểu dữ liệu)
!=	Khác giá trị
!==	Khác giá trị hoặc khác kiểu
&&	Và
||	Hoặc
!	Not

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

- Các kiểu dữ liệu trong JS
- String
- Number
- Boolean

  Var a = true; //hoặc false var b = 5<3 // b = false
- Object
- Undefined
- Array (là trường hợp đặc biệt của kiểu object)

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

- String
  - Để xuống dòng trong chuỗi sử dụng ký tứ "\" ở vị trí muốn xuống hoặc có thể dùng thẻ
 bên trong chuỗi.
  - Các ký tự đặc biệt được dùng trong chuỗi

Ký tự đặc biệt	Cách viết trong chuỗi
"	\"
'	\'
\	\\

  - Đếm ký tự trong chuỗi: var a = "học javascript";
    var num1 = a.length; //num1 = 14

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

❖String

Chuỗi	T	À	I		L	I	Ệ	U		H	Ọ	C		H	T	M	L
Chỉ số	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Vị trí	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

Phương thức / thuộc tính	Mô tả	Ví dụ
Length	Trả về số lượng ký tự	
toUpperCase()	Chuyển thường thành hoa	
toLowerCase()	Chuyển hoa thành thường	
Concat()	Nối các chuỗi với nhau	
indexOf()	Trả về chỉ số của chuỗi trùng khớp được tìm thấy đầu tiên	
lastIndexOf()	Trả về chỉ số của chuỗi trùng khớp sau cùng	
CharAt()	Trích xuất một ký tự trong chuỗi	
Substring()	Trích xuất một chuỗi con trong chuỗi	
Substr()	Trích xuất một chuỗi con trong chuỗi	
Replace()	Thay thế một nội dung nào đó trong chuỗi bằng nội dung mới	

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

❖Kiểu number

- Số nguyên: Độ chính xác 15 số
- Số thực: Độ chính xác 17 số (không tính dấu chấm)
- Có thể viết dưới dạng mũ (2e+6 = 2000000)
- Số có thể cộng với chữ nhưng không thể trừ, nhân, chia được
  - A = 10 + "Lê" // a = 10Lê
  - A = 10 – "lê" //a = NaN (Not a Number)
- Giá trị infinity (số lớn nhất – nằm ngoài khả năng đo đếm), -infinity.

❖Phương thức sử dụng cho số

❖Phương thức sử dụng cho số

Phương thức	Mô tả	Ví dụ
toString()	Chuyển giá trị về dạng chuỗi ký tự	Var a = 1999; Var b = a.toString(2) //chuyển a thành chuỗi nhị phân.
toExponential()	Trả về chuỗi biểu thức mũ của một số	Var c = a.toExponential() //c = 1.999e+3
toFixed()	Làm tròn một số đến số thập phân xác định	
Number()	Phân tích một giá trị để trả về một số tương ứng	Var a = Number(true) //a =1 Var b = Number(“199 3”) //b=NaN
ParseInt()	Phân tích một chuỗi để trả về một số nguyên tương ứng	
ParseFloat()	Phân tích một chuỗi để trả về một số thực tương ứng	

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

- Đối tượng Number trong JS, có các thuộc tính
  - Max_value: trả về giá trị lớn nhất có thể
  - Min_value
  - Positive_infinity: Trả về giá trị dương vô cực (Infinity)
  - Negative_infinity
  - Ví dụ: var a = Number.Max_value; //a = 1.7976….e+308
- Phương thức của thuộc tính Number

Phương thức	Mô tả	Ví dụ
isFinite()	Kiểm tra một giá trị nào đó có phải là số hữu hạn hay không	Var a = Number.isFinite(“1999”);//flase
isInteger()	Kiểm tra xem một giá trị nào đó có phải là số nguyên hay không	Var a = Number.isInteger(1999);//true
isNaN()	Kiểm tra giá trị nào đó có phải là NaN hay không.	Var a = Number.isNaN(1999); //false

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

❖Đối tượng Math (toán học) trong JS, có các phương thức:

Phương thức	Mô tả	Ví dụ
Abs()	Trả về giá trị tuyệt đối của sô	Math.abs(số)
Random()	Trả về giá trị ngẫu nhiên	Math.Random() //giá trị 0 tới 1
Sqrt()	Trả về giá trị căn bậc 2	
Pow()	Trả về lũy thừa của hai số xác định	Math.pow(x,y) //x mũ y
Ceil()	Làm tròn số lên một số nguyên gần nó nhất	
Floor()	Làm tròn một số xuống số nguyên gần nó nhất	
Round()	Làm tròn một số đến số nguyên gần nó nhất	//làm cả tròn lên (>=0.5) và tròn xuống
Max()	Trả về số lớn nhất trong danh sách	
Min()	Trả về số nhỏ nhất trong danh sách	

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

❖Object

- Là một tập hợp gồm những cái nên và mỗi tên chứa một giá trị.

```
    <!-- Kiểu object trong javascript -->
    <script>
      var SinhVien = {
      name:"Nhân",
      gender:"Nam",
      year:1999
  }
  document.write("<br>Giá trị của thuộc tính name là: " + SinhVien.name);
  document.write("<hr>");
  document.write("Giá trị của thuộc tính gender là: " + SinhVien.gender);
  document.write("<hr>");
  document.write("Giá trị của thuộc tính year là: " + SinhVien.year);
</script>

</body>
```

Giá trị của thuộc tính name là: Nhân

Giá trị của thuộc tính gender là: Nam

Giá trị của thuộc tính year là: 1999

- undefined
  - Biến chưa được gán giá trị sẽ có giá trị là undefined và kiểu dữ liệu là undefined.

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

❖Kiểu mảng

- Lưu trữ nhiều giá trị trong mảng, mỗi giá trị gọi là một phần tử mảng

```
<!-- Kiểu mảng -->
<script>
        var mobile = ["HTC","Nokia","SamSung"];
        document.write(mobile[0] + "<hr>");
        document.write(mobile[1] + "<hr>");
        document.write(mobile[2] + "<hr>");
    </script>
```

HTC

Nokia

SamSung

---

3. Câu lệnh trong JS
Kiểu dữ liệu trong JS

- Kiểu mảng
  - Giá trị trong mảng có thể là giống kiểu dữ liệu hoặc có thể là khác kiểu dữ liệu.
- Chỉ số mảng bắt đâù từ 0
- Có 3 cách gắn mảng:
  - C1: var dienthoai = [“Nokia”, “Samsung”, “Apple”]
  - C2: var dienthoai = new Array()

dienthoai[0] = “Nokia”

dienthoai[1] = “Samsung”

  - C3: var dienthoai = new Array()

dienthoai[“key”]=value; //Sử dụng key thay cho chỉ mục số

❖Cách hiển thị kiểu dữ liệu (có thể là 1 biến)

- Sử dụng toán tử typeof

```
<script>
        var a = typeof ""; //string
        var b = typeof "Lập Trình Web"; //string
        var c = typeof 1993; //number
        var d = typeof true; //boolean
        var e = typeof false; //boolean
        var f = typeof {name:"Nhân", gender:"Nam", year:1993};
//object
        var g = typeof undefined; //undefined
        var h = typeof ["HTC","Nokia","SamSung"]; //object
</script>
```

Javascript

1. Câu lệnh rẽ nhánh if
2. Câu lệnh rẽ nhánh switch
3. Vòng lặp For, while, do while

3. Câu lệnh trong JS
Câu lệnh điều kiện if … else

❖ Cú pháp

```
if(điều kiện 1){ //đoạn mã này sẽ được thực thi nếu điều kiện 1 là đúng
}
else if(điều kiện 2){ //đoạn mã này sẽ được thực thi nếu điều kiện 1 sai và điều kiện 2 là đúng
}
else{ //đoạn mã này sẽ được thực thi nếu tất cả những điều kiện trên là sai
}
```

3. Câu lệnh trong JS
Câu lệnh switch case

❖ Cú pháp

```
switch (giá trị){
   case trường hợp 1: //đoạn mã này sẽ được thực thi khi giá trị trùng khớp với trường hợp 1 break;
   case trường hợp 2: //đoạn mã này sẽ được thực thi khi giá trị trùng khớp với trường hợp 2 break;
   case trường hợp 3: //đoạn mã này sẽ được thực thi khi giá trị trùng khớp với trường hợp 3 break; ... ... ...
   default: //đoạn mã này sẽ được thực thi khi giá trị KHÔNG trùng khớp trường hợp nào cả break;
}
```

3. Câu lệnh trong JS
Vòng lặp

❖ Cú pháp vòng lặp for

```
for(biểu thức 1; biểu thức 2; biểu thức 3){ //Đoạn mã mà bạn muốn được thực thi nhiều lần }
```

❖For in: Dùng để lặp qua một lần các thuộc tính của một đối tượng.

```
<script>
var SinhVien = {
     name:“Lê Lung Linh",
     year:1999, gender:“Nữ",
     hometown:"Cần Thơ" }
for(x in SinhVien){
    document.write(SinhVien[x]);
   document.write("<hr>"); } </script>
```

3. Câu lệnh trong JS
Vòng lặp

- Cú pháp vòng lặp while

```
while(điều kiện){ //Đoạn mã mà bạn muốn thực thi
}
```

- Vòng lặp do…..while

```
do{ //Đoạn mã mà bạn muốn thực thi }while(điều kiện);
```

Javascript

1. Sử dụng hàm trong JS
2. Đối tượng trong JS

1. Sử dụng hàm (Function)

- Hàm là tập hợp nhiều câu lệnh để thực hiện một chức năng cụ thể.
- Một hàm có thể được gọi nhiều lần
- Hàm được chia làm hai loại cơ bản:
  - Không tham số.

```
function tên hàm() {
//Danh sách các câu lệnh của hàm }
```

  - Có tham số.

```
function tên hàm(tham số thứ nhất,
tham số thứ hai, tham số thứ ba,
....) { //Danh sách các câu lệnh của
hàm }
```

- Gọi hàm

```
tên hàm(giá trị tham số thứ nhất, giá trị
tham số thứ hai, giá trị tham số thứ ba,
....)
```

1. Sử dụng hàm (Function)

❖ Ví dụ:

```
<script>
        function GioiThieuBanThan(name, year){
            document.write("Tôi là " + name + " sinh năm " + year);
         }
        GioiThieuBanThan(); //Tôi là undefined sinh năm undefined
GioiThieuBanThan(“Lê Lung Linh"); //Tôi là Lê Lung Linh sinh năm undefined
GioiThieuBanThan(“Trần Trung Thực", 1989); //Tôi là Trần Trung Thực sinh năm
1989
GioiThieuBanThan("Tần Thúc Bảo", 1999); //Tôi là Tần Thúc Bảo sinh năm 1999
</script>
```

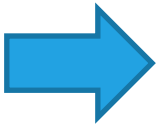
1. Sử dụng hàm (Function)

❖ Gọi hàm thông qua một sự kiện

```
<!-- Hàm thông qua sự kiện click -->
<button type="button" onclick="GioiThieu()">Bấm vào đây</button>

    <script>
        function GioiThieu(){
            alert("Bắt đầu học hàm trong javascript thông qua button lick.");
        }
    </script>
</body>
```

❖Sử dụng return trong hàm

```
<script>
    // Return trong hàm
    function number(){
        return (10*10 - 50);
    }

    var result_01 = number();

    var result_02 = 7 + number() - 30;

    var result_03 = "Hello: " + number();

document.write("<br>giá trị result_01 = " + result_01);
document.write("<br>giá trị result_02 = " + result_02);
document.write("<br>giá trị result_03 = " + result_03);
</script>
<br><br>
```



giá trị result_01 = 50
giá trị result_02 = 27
giá trị result_03 = Hello: 50

2. Sử dụng đối tượng

- Bên trong đối tượng có thể chứa nhiều cặp thuộc tính:giá trị
- Có thể khai báo đối tượng:
  - Cách 1: Định nghĩa và khởi tạo một đối tượng trong một lệnh

```
var person = {firstName:“Le", lastName:“Lung Linh", age:20, eyeColor:“black"};
```

  - Cách 2: Sử dụng từ khoá new.

```
var person = new Object();
person.firstName = " Le ";
person.lastName = " Lung Linh ";
person.age = 20;
person.eyeColor = “black";
```

2. Sử dụng đối tượng

- Sử dụng: document.getElementById("demo").innerHTML = person.firstName + " is " + person.age + " years old.";
- Sử dụng: var x = person;
  - X chính là person, khi thay đổi trong x như x.age = 10; có nghĩa trong person cũng thay đổi.
- person.firstName: Đây chính là thuộc tính của đối tượng. Ta có thể dùng person["firstname"]
- Sử dụng vòng lặp for..in để lấy hết các thuộc tính:

```
for (x in person) {
    txt += person[x];
}
```

- Thêm hoặc xoá thuộc tính của đối tượng dễ dàng:
  - person.nationality = "English"; //them thuộc tính nationality
  - delete person.age; // hoặc delete person["age"]; //xoá thuộc tính age

2. Đối tượng – Object methods

❖Có thể sử dụng phương thức trong đối tượng

```
// Khởi tạo một object:
var person = {
  firstName: “Le",
  lastName : “Lung Linh",
  id     : 1234,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
// Hiển thị dữ liệu trong Object
document.getElementById("demo").innerHTML = person.fullName();
```

```
var message = "Hello world!";
var x = message.toUpperCase();
```

toUpperCase() là hàm trong đối tượng String

❖“this” chỉ đối tượng person chứa hàm fullName.

Thêm method khi đã có đối tượng:

```
person.name = function () {
  return this.firstName + " " + this.lastName;      };
```

2. Đối tượng – Hiển thị đối tượng

❖Hiển thị đối tượng

var person = {name:“Linh", age:30, city:“Ha Noi"};

❖Cách 1: document.getElementById("demo").innerHTML = person;

❖ Cách 2: document.getElementById("demo").innerHTML = person.name + "," + person.age + "," + person.city;

❖Cách 3:
```
for (x in person) {
txt += person[x] + " ";
};

document.getElementById("demo").innerHTML = text;
```

❖Cách 4: var myArray = Object.values(person);

document.getElementById("demo").innerHTML = myArray;

Note: chưa đề cập tới JSON

2. Truy cập Object bằng get và set

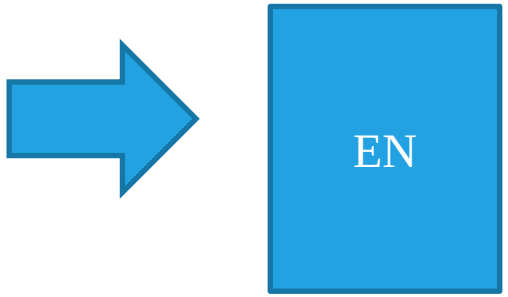
```
var person = {
  firstName: “Linh",
  lastName : “Le Lung",
  fullName : function() {
    return this.firstName + "
" + this.lastName;
  }
};

// Hiển thị dữ liệu sử dụng
method
document.getElementById("de
mo").innerHTML =
person.fullName();
```

```
var person = {
  firstName: “Linh",
  lastName : “Le Lung",
  get fullName() {
    return this.firstName + "
" + this.lastName;
  }
};

// Hiển thị dữ liệu sử dụng get:
document.getElementById("demo")
.innerHTML = person.fullName;
```

- *Truy cập hàm giống với thuộc tính*

2. Truy cập Object bằng get và set

```
var person = {
  firstName: “Linh",
  lastName : “Le Lung",
  language : "",
  set lang(lang) {
    this.language = lang.toUpperCase();
  }
};

// Đặt thuộc tính object sử dụng set
person.lang = "en";

// hiển thị dữ liệu của object
document.getElementById("demo").innerHTML =
person.language;
```



2. Truy cập Object bằng get và set

❖Sử dụng method Object.defineProperty() để dùng get và set:

```
// Define object
var obj = {counter : 0};

// Define setters
Object.defineProperty(obj, "reset", {
  get : function () {this.counter = 0;}
});
Object.defineProperty(obj, "increment", {
  get : function () {this.counter++;}
});
Object.defineProperty(obj, "decrement", {
  get : function () {this.counter--;}
});
Object.defineProperty(obj, "add", {
  set : function (value) {this.counter += value;}
});
Object.defineProperty(obj, "subtract", {
  set : function (value) {this.counter -= value;}
});
```

```
// Play with the counter:
obj.reset;
obj.add = 5;
obj.subtract = 1;
obj.increment;
obj.decrement;
```

JavaScript Getters and Setters

Perfect for creating counters:

4

2. Xây dựng Object

❖Person: chỉ một đối tượng là người tên là “Le Lung Linh”

⇒cần quản lý nhiều người khác????

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
this.eyeColor = eye;
}
```

“this” chỉ đối tượng chứa mã code

❖Sau đó khai báo cho nhiều object khác:

```
var myFather = new Person(“Le", “La", 50, "blue");
var myMother = new Person(“Nguyen", “Lan", 48, "green");
```

❖ Với mỗi đối tượng bên trong hàm tạo Person ta có thể thêm thuộc tính và method hoặc thay đổi giá trị bên trong mỗi object bình thường bình thường

- myFather.nationality = "English";
- myFather.name = function () {
  return this.firstName + " " + this.lastName;
  };

2. Sử dụng đối tượng (tt)

- Trong javascripts đã khởi tạo sẵn các đối tượng gốc:
  - var x1 = new Object(); // A new Object object
    var x2 = new String(); // A new String object
    var x3 = new Number(); // A new Number object
    var x4 = new Boolean(); // A new Boolean object
    var x5 = new Array(); // A new Array object
    var x6 = new RegExp(); // A new RegExp object
    var x7 = new Function(); // A new Function object
    var x8 = new Date(); // A new Date object
- Được viết ngắn gọn hơn:
  - var x1 = {}; // new object
    var x2 = ""; // new primitive string
    var x3 = 0; // new primitive number
    var x4 = false; // new primitive boolean
    var x5 = []; // new array object
    var x6 = /()/ // new regexp object
    var x7 = function(){}; // new function object

2. Thêm thuộc tính và method cho hàm tạo

❖Ta không thể thêm được thuộc tính và method cho hàm tạo

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
```

var myFather = new Person(“Le", “La", 50, "blue");
var myMother = new Person(“Nguyen", “Lan", 48, "green");

❖Person.nationality = "English";

Không viết như này

❖Person.prototype.nationality = "English";

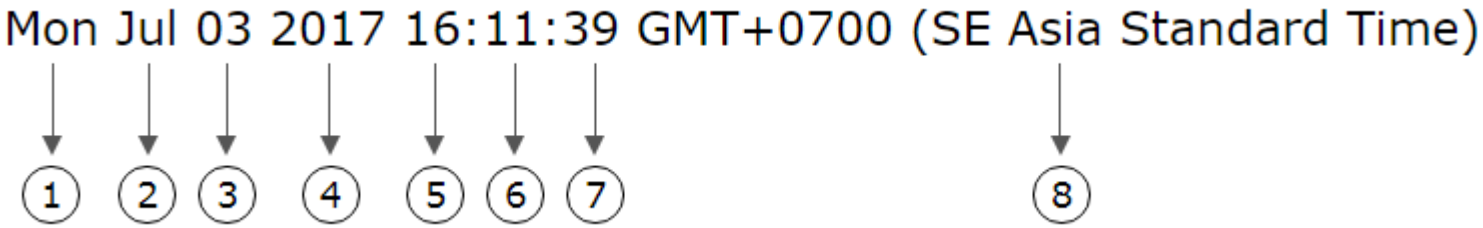
OK

Person.prototype.name = function() {
  return this.firstName + " " + this.lastName;
};

OK

2. Đối tượng ngày tháng

❖ Thứ tự hiển thị ngày tháng

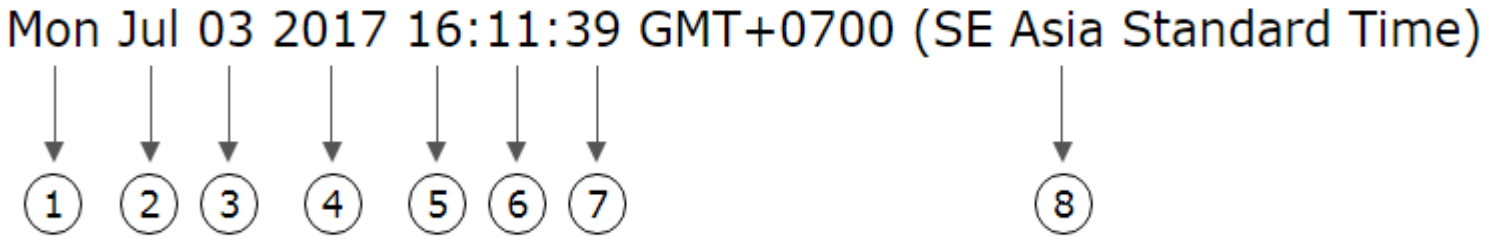


❖Cách khai báo

- Var date = new Date(); //biến date lưu trữ thời điểm hiện tại
- Var date1 = new date(mili giây);//ngày tháng bắt đầu 1/1/1970
- Var date2 = new date(year, month, day, hours, minutes, seconds, miliseconds)
  - Tháng: 0-11, ngày: 1-31, giờ: 0-23, phút, giây: 0-59
- New Date(chuỗi ngày tháng)
  - Các định dạng theo chuỗi date

2. Đối tượng ngày tháng

❖ Thứ tự hiển thị ngày tháng

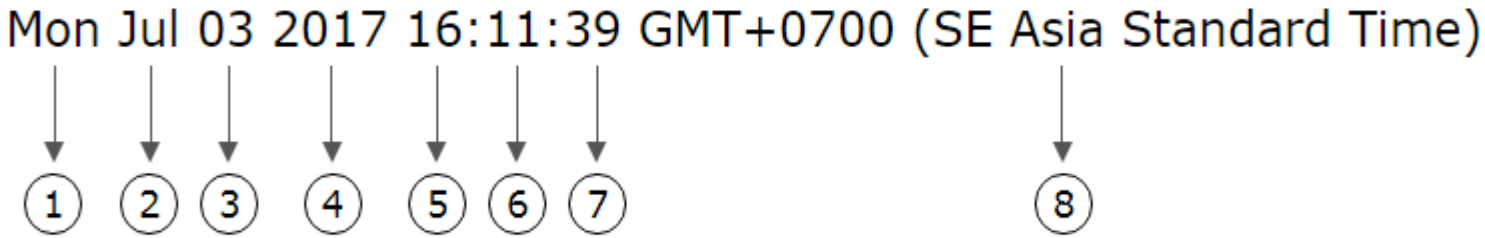


❖Cách khai báo

- Var date = new Date(); //biến date lưu trữ thời điểm hiện tại
- Var date1 = new date(mili giây);//ngày tháng bắt đầu 1/1/1970
- Var date2 = new date(year, month, day, hours, minutes, seconds, miliseconds)
  - Tháng: 0-11, ngày: 1-31, giờ: 0-23, phút, giây: 0-59
- New Date(chuỗi ngày tháng)
  - Các định dạng theo chuỗi date
  - Short date (03/08/2018), long date (Mar 08 2018),
  - full date(Thu Mar 08 2018), ISO date (2018-03-08)

2. Đối tượng ngày tháng

❖ Thứ tự hiển thị ngày tháng



❖Cách khai báo

- Var date = new Date(); //biến date lưu trữ thời điểm hiện tại
- Var date1 = new date(mili giây);//ngày tháng bắt đầu 1/1/1970
- Var date2 = new date(year, month, day, hours, minutes, seconds, miliseconds)
  - Tháng: 0-11, ngày: 1-31, giờ: 0-23, phút, giây: 0-59
- New Date(chuỗi ngày tháng) – có thể them được chuỗi thời gian.
  - Các định dạng theo chuỗi date
  - Short date (03/08/2018), long date (Mar 08 2018),
  - full date(Thu Mar 08 2018), ISO date (2018-03-08)

2. Đối tượng ngày tháng

❖ Các phương thức sử dụng trong kiểu Date – Lấy thông tin.

Phương thức	Mô tả
getDay()	Trả về ngày trong tuần *(Thứ)* của đối tượng ngày tháng *(có giá trị từ 0 - 6)*
getDate()	Trả về ngày trong tháng của đối tượng ngày tháng *(có giá trị từ 1 - 31)*
getMonth()	Trả về tháng của đối tượng ngày tháng *(có giá trị từ 0 - 11)*
getFullYear()	Trả về năm của đối tượng ngày tháng *(có giá trị từ 1000 - 9999)*
getHours()	Trả về giờ của đối tượng ngày tháng *(có giá trị từ 0 - 23)*
getMinutes()	Trả về phút của đối tượng ngày tháng *(có giá trị từ 0 - 59)*
getSeconds()	Trả về giây của đối tượng ngày tháng *(có giá trị từ 0 - 59)*
getMilliseconds()	Trả về mili giây của đối tượng ngày tháng *(có giá trị từ 0 - 999)*
getTime()	Trả về tổng số mili giây từ thời điểm 01/01/1970 00:00:00 đến thời điểm của đối tượng ngày tháng *(tính theo giờ tiêu chuẩn UTC)*

2. Đối tượng ngày tháng

❖ Các phương thức sử dụng trong kiểu Date – Thiết lập thông tin.

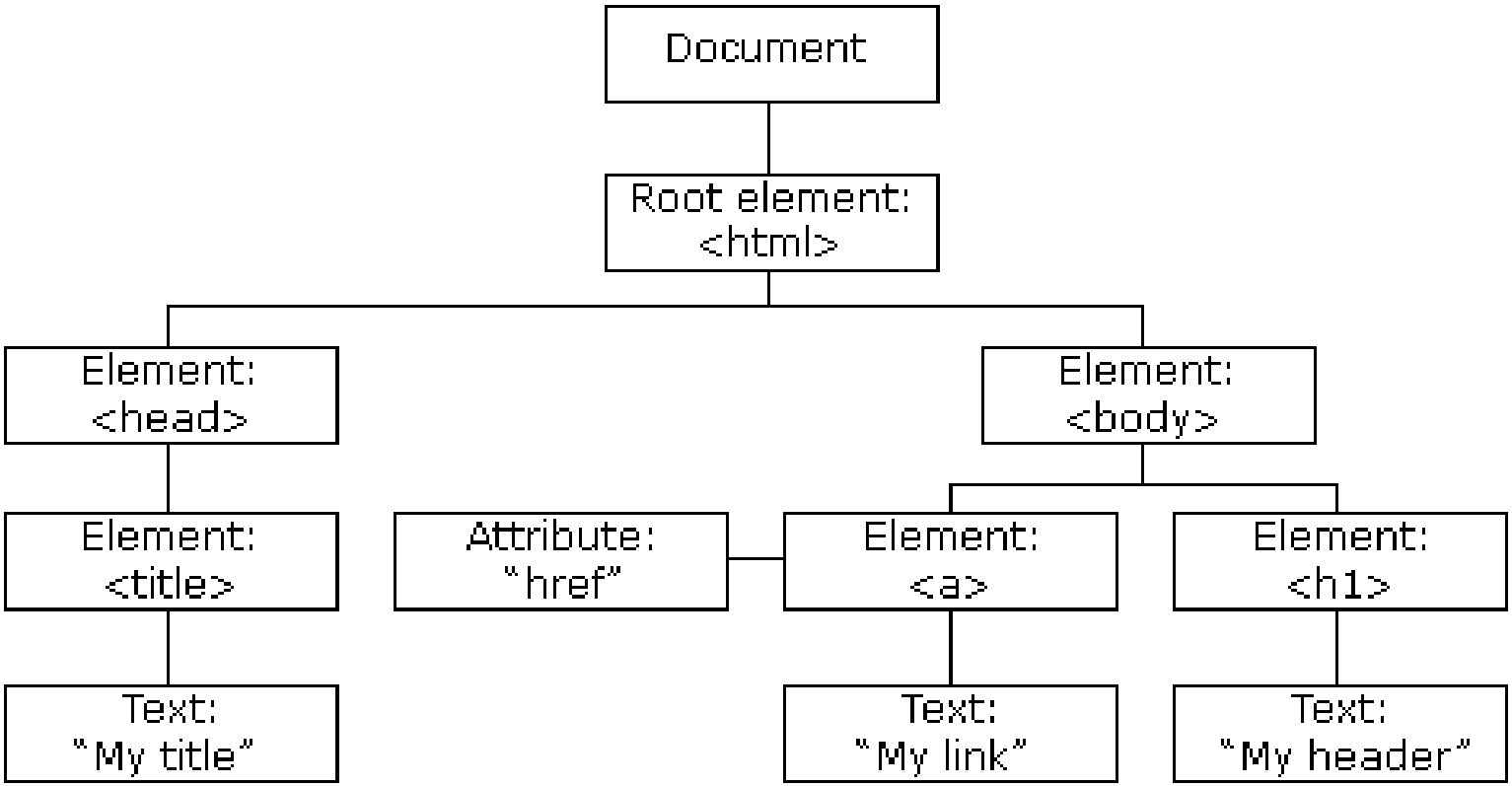
Phương thức	Mô tả
setDate()	Thiết lập lại ngày trong tháng của đối tượng ngày tháng
setMonth()	Thiết lập lại tháng của đối tượng ngày tháng
setFullYear()	Thiết lập lại năm của đối tượng ngày tháng
setHours()	Thiết lập lại giờ của đối tượng ngày tháng
setMinutes()	Thiết lập lại phút của đối tượng ngày tháng
setSeconds()	Thiết lập lại giây của đối tượng ngày tháng
setMilliseconds()	Thiết lập lại mili giây của đối tượng ngày tháng
setDate()	Thiết lập lại ngày trong tháng của đối tượng ngày tháng
setMonth()	Thiết lập lại tháng của đối tượng ngày tháng

JavaScript HTML DOM

- DOM method
- DOM Document
- DOM Elements
- DOM Event

Giới thiệu về DOM

- Khi trang web được tại, trình duyệt web khởi tạo DOM (Document Object Model)
- DOM khởi tạo cây Objects:



DOM Methods

- Các phương thức HTML DOM là các hành động tác động lên các phần tử HTML
- Thuộc tính DOM là các giá trị mà ta thiết lập hoặc thay đổi.

```
document.getElementById("demo").innerHTML = "Hello World!";
```

- innerHTML là thuộc tính thường xuyên được dùng, nó có thể dùng cho bất kỳ thẻ HTML nào bao gồm cả <html> và <body>

Đối tượng tài liệu HTML DOM

1. Tìm các phần tử HTML (có thể cả form)
   1. document.getElementById(*id*)
   2. document.getElementsByTagName(*name*)
   3. document.getElementsByClassName(*name*)
2. Thay đổi phần từ HTML
   1. *element*.innerHTML = *new html content (thuộc tính)*
   2. *element.attribute = new value (thuộc tính)*
   3. *element*.style.*property = new style (thuộc tính)*
   4. *element*.setAttribute*(attribute, value) (phương thức)*
3. Thêm và xoá phần tử
   1. document.createElement(*element*)
   2. document.removeChild(*element*)
   3. document.appendChild(*element*)
   4. document.replaceChild(*new*, *old*)
   5. document.write(*text*) //Ghi vào luồng đầu ra HTML

Đối tượng tài liệu HTML DOM

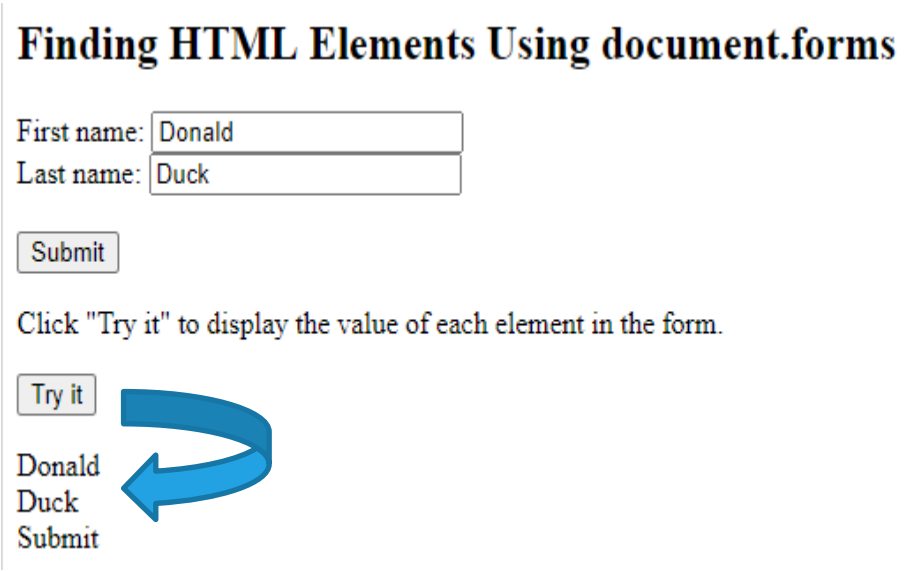
1. Thêm xử lý sự kiện
   1. document.getElementById(*id*).onclick = function(){*code*}
2. Tìm đối tượng HTML (tự thực hành)

document.anchors	document.documentURI	document.inputEncoding
document.applets	document.domain	document.lastModified
document.baseURI	document.domConfig	document.links
document.body	document.embeds	document.readyState
document.cookie	document.forms	document.referrer
document.doctype	document.head	document.scripts
document.documentElement	document.images	document.strictErrorChecking
document.documentMode	document.implementation	document.title
		document.URL

Một số ví dụ trong đối tượng

Ví dụ: Tìm đối tượng HTML bởi đối tượng HTML

```
<h2>Finding HTML Elements Using document.forms</h2>

<form id="frm1" action="/action_page.php">
  First name: <input type="text" name="fname" value="Donald"><br>
  Last name: <input type="text" name="lname" value="Duck"><br><br>
  <input type="submit" value="Submit">
</form>

<p>Click "Try it" to display the value of each element in the form.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var x = document.forms["frm1"];
  var text = "";
  var i;
  for (i = 0; i < x.length ;i++) {
    text += x.elements[i].value + "<br>";
  }
  document.getElementById("demo").innerHTML = text;
}
</script>
```



Một số ví dụ trong đối tượng

Ví dụ: Thay đổi giá trị của thuộc tính

document.getElementById(*id*).*attribute* = *new value;*

```
<img id="image" src="smiley.gif" width="160"
height="120">

<script>
document.getElementById("image").src =
"landscape.jpg";
</script>

<p>The original image was smiley.gif, but
the script changed it to landscape.jpg</p>
```

Một số ví dụ trong đối tượng

1. Thay đổi style HTML

   document.getElementById(*id*). style.*property* = *new style*

2. Sử dụng sự kiện

```
<html>
<body>

<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

</body>
</html>
```

Sự kiện trong JS

❖ Một số sự kiện hay sử dụng

Sự kiện	Mô tả
Onclick	Xảy ra khi người dùng click vào phần tử
Ondblclick	Xảy ra khi người dùng lick kép vào phần tử
Onmouseover	Xay ra khi người dùng di chuyển con trỏ vào phần tử
Onmouseleave Onmouseout	Xảy ra khi người dùng di chuyển con trỏ ra khỏi phần tử
Onkeyup	Xảy ra khi người dùng nhả phím ra
Oncopy	Xảy ra khi người dùng sao chép nội dung của phần tử
Oncut	Xảy ra khi người dùng cắt nội dung của phần tử
Onpaste	Xảy ra khi người dùng dán nội dung vào phần tử
Onchange	Xảy ra khi người dùng thay đổi giá trị của phần tử

Sự kiện trong JS (tiếp)

- Cách bắt sự kiện trong JS
  - Phần tử dùng để xảy ra sự kiện
  - Sự kiện sẽ xảy ra
  - Đoạn mã sẽ được thực thi khi sự kiện xảy ra.
- Cú pháp bắt sự kiện

```
<Tên-phần-tử Tên-sự-kiện="đoạn mã sẽ được thực thi khi sự kiện xảy ra">
```

Xử lý sự kiện

```
<body>

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
  id.innerHTML = "Ooops!";
}
</script>

</body>
```

```
<body>

<p>Click the button to display the date.</p>

<button onclick="displayDate()">The time is?</button>

<script>
function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
</script>

<p id="demo"></p>

</body>
```

Xử lý sự kiện

```
<body>

<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>

<script>
function mOver(obj) {
  obj.innerHTML = "Thank You"
}

function mOut(obj) {
  obj.innerHTML = "Mouse Over Me"
}
</script>

</body>
```

JavaScript Forms

JavaScript Form Validation

JavaScript Validation API

Biểu thức chính quy

1. Kiểm tra Form

- Có thể dùng Javascript để Kiểm tra form trước khi gửi lên server
- Kiểm tra thường đi kèm với các sự kiện dùng trong form
- Sử dụng kèm với "Biểu thức chính quy"
- Một số kiểm tra điều kiện đã có sẵn ở html (thuộc tính required) hay sử dụng pseudo selectors trong CSS
  - :disabled
  - :invalid
  - :optional
  - :required
  - :valid
  - …

2. Kiểm tra Form sử dụng API có sẵn

- Sử dụng một số method đã hỗ trợ sẵn trong JS
  - checkValidity(): Trả về true nếu nếu element chứa dữ liệu hợp lệ
  - setCustomValidity(): Đặt thuộc tính validationMessage của element input.

```
<input id="id1" type="number" min="100" max="300" required>
<button onclick="myFunction()">OK</button>

<p id="demo"></p>

<script>
function myFunction() {
  var inpObj = document.getElementById("id1");
  if (!inpObj.checkValidity()) {
    document.getElementById("demo").innerHTML = inpObj.validationMessage;
  }
}
</script>
```

Không hiển thị lỗi khi nhập đúng trong khoảng 100-300
Báo lỗi khi khác.

Kiểm tra Form sử dụng API có sẵn

- Một số thuộc tính liên quan tới tính hợp lệ của dữ liệu trong thuộc tính validity:

Property	Description
customError	Set to true, if a custom validity message is set.
patternMismatch	Set to true, if an element's value does not match its pattern attribute.
rangeOverflow	Set to true, if an element's value is greater than its max attribute.
rangeUnderflow	Set to true, if an element's value is less than its min attribute.
stepMismatch	Set to true, if an element's value is invalid per its step attribute.
tooLong	Set to true, if an element's value exceeds its maxLength attribute.
typeMismatch	Set to true, if an element's value is invalid per its type attribute.
valueMissing	Set to true, if an element (with a required attribute) has no value.
valid	Set to true, if an element's value is valid.

Kiểm tra Form sử dụng API có sẵn

❖Ví dụ

```
<body>

<p>Enter a number and click OK:</p>

<input id="id1" type="number" max="100">
<button onclick="myFunction()">OK</button>

<p>If the number is greater than 100 (the input's max attribute), an error message will be
displayed.</p>

<p id="demo"></p>

<script>
function myFunction() {
  var txt = "";
  if (document.getElementById("id1").validity.rangeOverflow) {
    txt = "Value too large";
  } else {
    txt = "Input OK";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
```

Biểu thức chính quy trong JS

- Cách tạo một biểu thức chính quy
  - Cách 1: Biểu thức chính quy thuần `var re = /ab+c/;`
  - Cách 2: Tạo một đối tượng RegExp

```
var re = new RegExp("ab+c");
```

- Cách viết một biểu thức chính quy
  - Đơn giản: /abc/ //Tìm các đoạn abc theo đúng thứ tự chuỗi
  - /ab*c/ //tìm đoạn có chứa 1 ký tự a + có hoặc nhiều b và 1 ký tự c

Biểu thức chính quy trong JS

It.kma

Bảng mô tả các ký tự đặc biệt

TT	Biểu thức chính quy	Mô tả
1	.	Khớp (match) với bất kỳ ký tự nào
2	^regex	Biểu thức chính quy phải khớp tại điểm bắt đầu
3	regex$	Biểu thức chính quy phải khớp ở cuối dòng.
4	[abc]	Thiết lập định nghĩa, có thể khớp với a hoặc b hoặc c.
5	[abc][vz]	Thiết lập định nghĩa, có thể khớp với a hoặc b hoặc c theo sau là v hay z.
6	[^abc]	Khi dấu ^ xuất hiện như là nhân vật đầu tiên trong dấu ngoặc vuông, nó phủ nhận mô hình. Điều này có thể khớp với bất kỳ ký tự nào ngoại trừ a hoặc b hoặc c.
7	[a-d1-7]	Phạm vi: phù hợp với một chuỗi giữa a và điểm d và con số từ 1 đến 7.
8	X|Z	Tìm X hoặc Z.
9	XZ	Tìm X và theo sau là Z.
10	$	Kiểm tra kết thúc dòng.

11	\d	Số bất kỳ, viết ngắn gọn cho [0-9]
12	\D	Ký tự không phải là số, viết ngắn gon cho [^0-9]
13	\s	Ký tự khoảng trắng, viết ngắn gọn cho [\t\n\x0b\r\f]
14	\S	Ký tự không phải khoản trắng, viết ngắn gọn cho [^\s]
15	\w	Ký tự chữ, viết ngắn gọn cho [a-zA-Z_0-9]
16	\W	Ký tự không phải chữ, viết ngắn gọn cho [^\w]
17	\S+	Một số ký tự không phải khoảng trắng (Một hoặc nhiều)
18	\b	Ký tự thuộc a-z hoặc A-Z hoặc 0-9 hoặc _, viết ngắn gọn cho [a-zA-Z0-9_].
19	*	Xuất hiện 0 hoặc nhiều lần, viết ngắn gọn cho {0,}
20	+	Xuất hiện 1 hoặc nhiều lần, viết ngắn gọn cho {1,}
21	?	Xuất hiện 0 hoặc 1 lần, ? viết ngắn gọn cho {0,1}.
22	{X}	Xuất hiện X lần, {}
23	{X,Y}	Xuất hiện trong khoảng X tới Y lần.
24	*?	* có nghĩa là xuất hiện 0 hoặc nhiều lần, thêm ? phía sau nghĩa là tìm kiếm khớp nhỏ nhất.

- Ngoặc tròn bao quanh bất kỳ phần tử nào của biểu thức chính quy sẽ khiến phần kết quả so khớp được nhớ. Mỗi lần nhớ, chuỗi con có thể được gọi lại để sử dụng.
- Ví dụ: /Chapter (\d+).\d*/ so khớp 'Chapter' theo sau bởi 1 hoặc nhiều ký tự số, sau nó là dâu "," thập phân, cuối cùng có thể là 0 hoặc nhiều số đầu tiên được khớp.

Làm việc với biểu thức chính quy

❖ Biểu thức chính quy được sử dụng với phương thứ test và exec của lớp RegExp hoặc match, replace, search, split của String.

Phương thức	Mô tả
exec	Một phương thức của RegExp dùng để tìm kiếm chuỗi phù hợp với mẫu so khớp. Nó trả về một mảng chứa kết quả tìm kiếm.
test	Một phương thức của RegExp dùng để kiểm tra mẫu có khớp với chuỗi hay không. Nó trả về giá trị true hoặc false.
match	Một phương thức của chuỗi dùng để tìm kiếm chuỗi phù hợp với mẫu so khớp. Nó trả về một mảng chứa kết quả tìm kiếm hoặc null nếu không tìm thấy.
search	Một phương thức của chuỗi dùng để tìm kiếm chuỗi phù hợp với mẫu so khớp và trả về vị trí của chuỗi đó hoặc -1 nếu không tìm thấy.
replace	Một phương thức của chuỗi dùng để tìm kiếm một chuỗi theo mẫu so khớp và thay thế chuỗi con được khớp với một chuỗi thay thế.
split	Một phương thức của chuỗi dùng một biểu mẫu chính quy hoặc một chuỗi bất biến để ngắt chuỗi đó thành một mảng các chuỗi con.

❖Ví dụ tìm chuỗi phù hợp theo mẫu so khớp:

```
var myRe = /d(b+)d/g; var myArray = myRe.exec("cSdbbdbsbz");
```

Hoặc:

```
var myArray = /d(b+)d/g.exec("cdbbdbsbz");
```

Biểu thức chính quy trong JS
Làm việc với biểu thức chính quy

- var myRe = new RegExp("d(b+)d", "g");
  var myArray = myRe.exec("cdbbdbsbz");
- Khi so khớp thành công sẽ trả về một mảng kết quả

Đối tượng	Thuộc tính hoặc chỉ số	Mô tả	Trong vd này
`myArray`		Chuỗi kết quả so khớp và toàn bộ chuỗi con được nhớ.	`["dbbd", "bb"]`
	`index`	Vị trí của chuỗi tìm thấy, đếm từ 0	`1`
	`input`	Chuỗi gốc được nhập vào	`"cdbbdbsbz"`
	`[0]`	Những kí tự cuối cùng được khớp.	`"dbbd"`
`myRe`	`lastIndex`	Vị trí nơi mà bắt đầu so khớp lần sau. (Thuộc tính này chỉ được gán nếu biểu thức chính quy sử dụng lựa chọn g, được mô tả trong Advanced Searching With Flags).	`5`
	`source`	Chuỗi biểu thức chính quy, được cập nhật tại thời điểm biểu thức được tạo, không phải tại lúc thực thi.	`"d(b+)d"`

- var myRe = /d(b+)d/g;
  var myArray = myRe.exec("cdbbdbsbz");
  console.log("The value of lastIndex is " + myRe.lastIndex);

Javascript

1. Xử lý lỗi
2. Thuật ngữ hoisting trong JS
3. Chế độ nghiêm ngặt Strict Mode

4. Câu lệnh xử lý lỗi trong JS

- Sử dụng try….catch
  - Tránh xảy ra lỗi mặc định: khi JS gặp lỗi thì việc thực thi sẽ kết thúc.

```
<script>
    document.write("<p>Tài liệu học HTML</p>");
    try{
        document.write("<p>Tài liệu học CSS</p>");
        aaaleert("<p>Tài liệu học JavaScript</p>");
        document.write("<p>Tài liệu học MySQL</p>");
        document.write("<p>Tài liệu học PHP</p>");
    }catch(err){
        document.write("<p>Bên trong lệnh try chứa câu lệnh bị lỗi</p>");
        document.write("<p>Bên trong lệnh try chứa câu lệnh bị lỗi</p>");
        document.write("<p>Bên trong lệnh try chứa câu lệnh bị lỗi</p>");
    }
    document.w      "<p>Tài liệu học PHP</p>");
</script>
```

Thực thi tới đây.

Dòng lệnh lỗi, chạy bên trong catch.

Biến err để lấy lỗi khi muốn đọc xem JS bị lỗi gì

4. Câu lệnh xử lý lỗi trong JS (tiếp)

- Sử dụng lệnh throw
  - Thường được viết bên trong lệnh try
  - Có chức năng ném ra 1 lỗi (throw thực thi => 1 lỗi được tạo ra).

```
<script>
    function KiemTraDuLieu() {
        document.getElementById("demo").innerHTML = "";
        var number = document.getElementById("number").value;
        try{
            if(number == ""){
                throw "Bạn chưa nhập giá trị";
            }else if(isNaN(number)){
                throw "Giá trị bạn nhập không phải là một số";
            }else if(number < 1){
                throw "Giá trị vừa nhập nhỏ hơn giá trị cho phép";
            }else if(number > 10){
                throw "Giá trị vừa nhập lớn hơn giá trị cho phép
            }
            document.getElementById("demo").innerHTML = "Hợp lệ";
        }
        catch(err){
            document.getElementById("demo").innerHTML = err;
        }
    }
</script>
```

Gặp throw chương trình sẽ dừng lại, in giá trị bên cạnh throw

4. Câu lệnh xử lý lỗi trong JS (tiếp)

❖ Sử dụng lệnh finally

- Thường đặt phía sau try hoặc try…catch
- Có chức năng ném ra 1 lỗi (throw thực thi => 1 lỗi được tạo ra).

```
<p>Nhập một số có giá trị trong khoảng từ 1 đến 10</p>

<input type="text" id="number">

<button type="button" onclick="KiemTraDuLieu()">Kiểm tra</button>

<p id="demo"></p>

<script>
    function KiemTraDuLieu() {
        document.getElementById("demo").innerHTML = "";
        var number = document.getElementById("number").value;
        try{
            if(number == ""){
                throw "Bạn chưa nhập giá trị";
            }else if(isNaN(number)){
                throw "Giá trị bạn nhập không phải là một số";
            }else if(number < 1){
                throw "Giá trị vừa nhập nhỏ hơn giá trị cho phép";
            }else if(number > 10){
                throw "Giá trị vừa nhập lớn hơn giá trị cho phép";
            }
            document.getElementById("demo").innerHTML = "Hợp lệ";
        }
        catch(err){
            document.getElementById("demo").innerHTML = err;
        }
        finally{
            document.getElementById("number").value = "finally sẽ chạy cuối cùng";
        }
    }
</script>
```

Nhập một số có giá trị trong khoảng từ 1 đến 10

finally sẽ chạy cuối cùng | Kiểm tra

Giá trị vừa nhập lớn hơn giá trị cho phép

4. Câu lệnh xử lý lỗi trong JS (tiếp)

❖Đối tượng lưu thông tin lỗi

- Lệnh catch bắt buộc phải có tham số, tham số này dùng để lưu trữ một đối tượng. Đối tượng này chứa các thông tin lỗi của câu lệnh bị lỗi trong try.
- Đối tượng lưu thông tin lỗi có 2 thuộc tính:
  - Name: trả về tên loại lỗi của câu lệnh lỗi
  - Message: trả về thông báo lỗi của mỗi lệnh lỗi
- Các loại lỗi cơ bản
  - ReferenceError: Sử dung biến chưa được khai báo
  - URIError: Sử dụng các ký tự không hợp lệ trong các hàm xử lý URI
  - TypeError: Kiểu dữ liệu nằm ngoài mong đợi
  - RangeError: Tham số nằm ngoài phạm vi giá trị cho phép
  - SyntaxError: Câu lệnh sai cú pháp

4. Câu lệnh xử lý lỗi trong JS (tiếp)

```
<script>
    try{
        document.write(text);
    }catch(err){
        document.write(err);
        document.write("<br>");
        document.write("Loại lỗi: " + err.name);
        document.write("<br>");
        document.write("Thông báo lỗi: " +
err.message);
    }
</script>
```

```
ReferenceError: text is not defined
Loại lỗi: ReferenceError
Thông báo lỗi: text is not defined
```

5. Thuật ngữ hoisting trong JS

- Hoisting: là hành động mặc định của JS, nó sẽ di chuyển những câu lệnh khai báo tên biến lên vị trí đầu tiên trong phạm vi (đầu tập tin hoặc đầu hàm hiện tại).
- Ví dụ

```
var x = 5; // Initialize x
var y = 7; // Initialize y
elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y
```

```
var x = 5; // Initialize x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y

var y = 7; // Initialize y
```

5. Thuật ngữ hoisting trong JS

- Hoisting: là hành động mặc định của JS, nó sẽ di chuyển những câu lệnh khai báo tên biến lên vị trí đầu tiên trong phạm vi (đầu tập tin hoặc đầu hàm hiện tại).
- Ví dụ

```
var x = 5; // Initialize x
var y = 7; // Initialize y
elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y
```

5 7

```
var x = 5; // Initialize x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y

var y = 7; // Initialize y
```

x is 5 and y is undefined

5. Thuật ngữ hoisting trong JS

- Hoisting: là hành động mặc định của JS, nó sẽ di chuyển những câu lệnh khai báo tên biến lên vị trí đầu tiên trong phạm vi (đầu tập tin hoặc đầu hàm hiện tại).
- Ví dụ

```
<script>
var x = 5; // Initialize x
var y;     // Declare y
elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y
y = 7;     // Assign 7 to y
</script>
```

5 undefined

- Nếu đang ở trong hàm, ta sử dụng biến chưa được khai báo thì biến đó sẽ trở thành biến toàn cục
- JS cho chép sử dụng gán 1 biến chưa được khai báo.

6. Chế độ nghiêm ngặt Strict Mode

- Dù chưa khai báo x, và y nhưng đoạn mã này vẫn chạy.

```
x = 5; // Initialize x
y = 7; // Initialize y
elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y
```

- Sử dụng chế độ nghiêm ngặt Strict Mode để tránh tình trạng viết code ẩu.
  - Cú pháp: sử dụng “use strict” ở dòng đầu JS

```
<script>

    "use strict";

    str1 = "Lập Trình Web"; //Câu lệnh này bị lỗi

    function Hello(){
        str2 = "Xin chào JavaScript"; //Câu lệnh này bị lỗi
    }

</script>
```

6. Chế độ nghiêm ngặt Strict Mode

Những điều không được phép trong chế độ nghiêm ngặt

- Không được gán giá trị cho một biến (mảng, đối tượng,…) khi nó chưa được khai báo
- Không được phép xóa: biến, mảng, đối tượng, hàm, … (delete a;)
- Trong việc khai báo hàm, các tham số của hàm không được trùng tên nhau
- Một giá trị là số không được bắt đầu bằng số 0 (var year =01999; // lỗi)
- Không được cập nhật giá trị của một "thuộc tính chỉ có thể đọc"
- Không được sử dụng các từ sau để đặt tên cho biến
  - eval, arguments, implements, interface, let, package, private, protected, public, static, yield.

```
<script>
   "use strict"; var SinhVien = {name:"Lê Lung Linh", year:1999};
   Object.defineProperty(SinhVien, "year", {writable:false}); SinhVien.year =
2017; //Câu lệnh này bị lỗi
   document.write(SinhVien.year);
</script>
```

Bài tập cuối chương JS