**JAVASCRIPT**

**I. JavaScript là gì?**

- JavaScript, viết tắt là JS là một ngôn ngữ lập trình thông dịch trên được phát triển từ ý tưởng lập trình dựa trên nguyên mẫu. JS được sử dụng rộng rãi cho các trang web (phía người dùng) cũng như phía máy chủ (với NodeJS). Phần mở rộng thường là .js

*\*NodeJS: là một cách để JavaScript chạy trên phía Server.*

- Mã nguồn JS thường được nhúng hoặc tích hợp vào tài liệu HTML. Khi trang web được tải xong, trình duyệt sẽ thông dịch và thực hiện các mã này, hoặc khi người dùng tác động một sự kiện nào đó lên trình duyệt.

- Overview về JS:

+ Là ngôn ngữ có tính chặt chẽ dữ liệu thấp.

+ Nhận diện chữ hoa chữ thường (case sensitive)

+ String: đóng trong nháy đơn hoặc nháy đôi đều được.

+ Number: không nằm trong nháy, có thể nhận tất cả các giá trị số kể cả hexa.

+ Hỗ trợ boolean với hai giá trị true false

+ Không nhất thiết phải dùng dấu “;” cuối lệnh nhưng nên làm vậy.

+ Bỏ qua nhiều dấu cách, ví dụ: a = 1 thì không lỗi.

+ Comment: một dòng //, nhiều dòng: /\* \*/

+ Keyword không được dùng cho tên biến hay hàm, đa số giống C#, Java

+

- Tác dụng của JS:

+ Làm cho tài liệu HTML trở nên sinh động hơn.

+ Phản ứng lại một sự kiện nào đó từ phía người dùng.

+ Đọc hoặc thay đổi nội dung trong phần tử HTML.

+ Kiểm tra dữ liệu.

+ Phát hiện các loại trình duyệt khác nhau.

+ Tạo ra tập tin cookie lưu trữ và truy xuất thông tin trên máy tính của người dùng.

*\*Nhúng JS vào tài liệu HTML: (nên nhúng ở dòng cuối của thẻ body hoặc dùng document ready)*

- Để nhúng một đoạn JS vào tài liệu HTML, ta chỉ cần đưa câu lệnh đó vào thẻ <script> như sau.

<script type=”text/javascript”> // dùng để phân biệt với VBScript.

…

</script>

- Để tài liệu HTML nhúng JS có thể hoạt động bình thường trên các trình duyệt không hỗ trợ JS, cần thêm comment của HTML vào phần nội dung trong thẻ script.

<script type=”text/javascript”> // dùng để phân biệt với VBScript.

<!--

…

-->

</script>

- Để gọi hàm JS trong HTML, ta có thể làm như sau: (tạo ra một script chứa hàm và gọi nó với một hành động bắt sự kiện).

Đối với trạng thái của tài liệu:

<**html**>

<**head**>

<**style** type=*"text/css"*> <!-- Cho code của CSS -->

</**style**>

<**script** type=*"text/javascript"*> <!-- Cho code của JS -->

**function** message() {

alert("Hello");

}

</**script**>

</**head**>

<**body** onLoad="message();">

</**body**>

</**html**>

Đối với trạng thái của các phần tử khác trong HTML: (Ví dụ cho phần tử link)

<**html**>

<**head**>

<**style** type=*"text/css"*> <!-- Cho code của CSS -->

</**style**>

<**script** type=*"text/javascript"*> <!-- Cho code của JS -->

**function** message() {

alert("Hello");

}

</**script**>

</**head**>

<**body**>

<**a** href=*"#"* onClick="message();">Click here</**a**>

</**body**>

</**html**>

- Vị trí của JS trong file HTML:

+ Trong cặp thẻ <head> </head>:

<**html**>

<**head**>

<**style** type=*"text/css"*>

</**style**>

</**head**>

<**body**>

<**script** type=*"text/javascript"*>

**function** message() {

alert("Hello World!");

}

</**script**>

<**a** href=*"#"* onClick="message();">Click here</**a**>

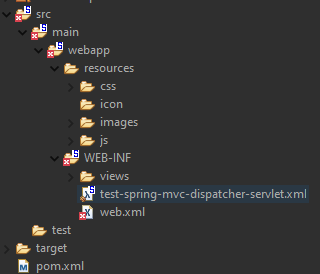
</**body**>

</**html**>

Can not find the tag library descriptor for "http://java.sun.com/jsp/jstl/core"

- Trong Spring MVC, để tích hợp được file CSS và file JS vào trong JSP thì cần phải khai báo và tổ chức thư mục như sau:

+ Thư mục:



+ file .jsp: (dùng tag của jstl)

<%@ **page** language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"* session=*"false"*%>

<%@ **taglib** prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<!**DOCTYPE** html>

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Insert title here</**title**>

<**link** rel=*"stylesheet"*

href=*"*${pageContext.request.contextPath}*/resources/css/style.css"*>

<**script** type=*"text/javascript"*

src=*"*${pageContext.request.contextPath}*/resources/js/sample.js"*></**script**>

</**head**>

<**body**>

<**p**>Nothing</**p**>

<**h2**>Nothing</**h2**>

<**a** href=*"#"* onclick="doSomeWork();">Click</**a**>

<**img**

src=*"*${pageContext.request.contextPath}*/resources/images/relogo.png"* />

</**body**>

</**html**>

+ file .css chứa trong thư mục css, file.js chứa trong thư mục js, file hình ảnh chứa trong thư mục images.

+ file .css:

body {

background-color: *lightblue*;

}

h2 {

color: *Blue*;

text-align: *right*;

}

p {

font-family: *verdana*;

font-size: *20px*;

font-color: *Red*;

}

+ file: .js:

**function** doSomeWork() {

alert("wow js is working in spring mvc....");

}

+ file …-dispatcher-servlet.xml: cần có phần <mvc:resources mapping> …

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd"*>

<**context:component-scan**

base-package=*"com.dongnv.controller"* />

<**bean**

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<**property** name=*"prefix"* value=*"/WEB-INF/views/"*></**property**>

<**property** name=*"suffix"* value=*".jsp"*></**property**>

</**bean**>

<**mvc:resources** mapping=*"/resources/\*\*"* location=*"/resources/"* />

<**mvc:annotation-driven** />

</**beans**>

=> Kết quả:



*\*Các thành phần cú pháp chính của JavaScipt:*

- Khoảng trắng: Dấu cách, tab và enter sử dụng bên ngoài một chuỗi ký tự ảnh hưởng trực tiếp tới ý nghĩa của code.

- Chú thích trong JS: Cú pháp comment trong JS giống với C++, // và /\* … \*/

- Biến: Người dùng có thể chỉ cần gán giá trị cho biến và sử dụng mà không nhất thiết phải khai báo biến. Muốn khai báo biến trong JS có thể dùng tới từ khoá let và const (có thể và không thể thay đổi giá trị).

- Toán tử.

+ Toán tử số học: cộng (+), trừ (-), nhân (\*), chia (/), chia lấy dư (%).

var result;

result = 2 + 4; // result = 6

result = 2 - 4; // result = -2

result = 2 \* 4; // result = 8

result = 2 / 4; // result = 0.5

result = 2 % 4; // result = 2

+ Toán tử tăng giảm: ++ và - - chỉ thực hiện trên một toán hạng duy nhất, làm tăng hoặc giảm giá trị toán hạng đi 1.

var x = 2;

var y;

y = x++; // x = 3, y = 2

y = ++x; // x = 3, y = 3

y = x--; // x = 1, y = 2

y = --x; // x = 1, y = 1

+ Toán tử quan hệ: dùng để so sánh hai toán hạng hoặc hai biểu thức, trả lại các giá trị true hoặc false, gồm các phép toán: == (bằng nhau), != (khác), === (bằng nhau và cùng loại), !== (khác nhau và khác loại)¸ >, <, >=, <=.

var result;

result = 3 == "3"; // result = true

result = 3 != 3; // result = false

result = 3 === "3"; // result = false

result = 3 !== "3"; // result = true

result = 3 > 4; // result = false

result = 3 < 4; // result = true

result = 3 >= 3; // result = true

result = 3 <= 4; // result = true

+ Toán tử luận lý: Là các toán tử thực hiện các phép toán logic trên hai toán hạng, thuộc về toán tử quan hệ, trả về các giá trị là true hoặc false, bao gồm các phép toán: &&, || và !.

var x = 2, y = 5;

var result;

result = (x == 3) && (y == 5); // result = false

result = (x == 3) || (y == 5); // result = true

result = !(x == 3); // result = true

+ Toán tử đặc biệt: giống toán tử 3 ngôi trong C++ có cú pháp như sau:

điều\_kiện ? biểu\_thức\_đúng: biểu\_thức\_sai;

- Câu lệnh điều khiển:

+ Câu lệnh if … else:

if (biểu\_thức\_1) {

khối lệnh được thực hiện nếu biểu thức 1 đúng;

}

else if (biểu\_thức\_2) {

khối lệnh được thực hiện nếu biểu thức 1 sai và biểu thức 2 đúng;

}

else {

khối lệnh được thực hiện nếu cả hai biểu thức trên đều sai;

}

+ Câu lệnh: switch … case:

switch (biểu\_thức\_điều\_kiện) {

case kết\_quả\_1:

khối lệnh cần thực hiện nếu biểu\_thức\_điều\_kiện bằng kết\_quả\_1;

break;

case kết\_quả\_2:

khối lệnh cần thực hiện nếu biểu\_thức\_điều\_kiện bằng kết\_quả\_2;

break;

default:

khối lệnh cần thực hiện nếu biểu\_thức\_điều\_kiện cho ra một kết quả khác;

break;

}

- Vòng lặp:

+ while:

while (biểu\_thức\_điều\_kiện) {

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

}

+ do … while:

do {

khối lệnh;

}

while (biểu\_thức\_điều\_kiện);

+ for:

for (biểu\_thức\_khởi\_tạo; biểu\_thức\_điều\_kiện; biểu\_thức\_thay\_đổi\_giá\_trị) {

Khối lệnh cần lặp;

}

+ for … in:

for (biến in đối\_tượng) {

khối lệnh cần thực hiện, có thể sử dụng đối\_tượng[biến] để truy cập từng thuộc tính (phần tử) của đối tượng;

}

- Hàm: Là một khối các câu lệnh với danh sách gồm một hoặc nhiều đối số (có thể không có đối số) và có tên hàm (trong JS không nhất thiết phải có tên).

//ECMAScript 6 trở đi

const tên\_hàm = (đối\_số\_1, đối\_số\_2)=>{

//câu lệnh

}

//Thực thi

tên\_hàm(1, 2); // Gọi hàm tên\_hàm với hai đối số 1 và 2 ứng với đối\_số\_1 và đối\_số\_2

tên\_hàm(1); // Gọi hàm tên\_hàm với đối\_số\_1 có giá trị 1, đối\_số\_2 có giá trị undefined

Trong JS không nhất thiết phải thực thi hàm với đủ đối số, khi đó giá trị các đối số không được gán sẽ mang giá trị undefine.

Có hai cách truyền tham số cho hàm: truyền giá trị cho các kiểu dữ liệu cơ bản, hoặc truyền tham chiếu cho các đối tượng.

Trong JS, hàm là đối tượng hạng nhất, tất cả đều là đối tượng của nguyên mẫu function, hàm có thể được tạo và gán như bất kỳ một đối tượng nào khác, và cũng có thể được dùng làm đối số cho các hàm khác. Điều này làm cho JS trở thành ngôn ngữ hỗ trợ hàm cấp độ cao.

Array.prototype.fold =

function (value, functor) {

var result = value;

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

result = functor(result, this[i]);

}

return result;

}

var sum = [1,2,3,4,5,6,7,8,9,10].fold(0, function (a, b) { return a + b; });

Hàm trong CSS là đối tượng nên có thể khởi tạo một cách ẩn danh:

function() { thân hàm; }

Ví dụ hàm ẩn danh:

document.onkeypress = function(e) {

alert("Bạn vừa nhấn một phím trên bàn phím");

}

- Mảng:

- Đối tượng:

- Quản lý lỗi:

try {

Khối lệnh cần thực hiện có thể gây lỗi;

}

catch (error) {

Khối lệnh cần thực hiện trong trường hợp có lỗi;

}

finally {

Khối lệnh luôn được thực hiện;

}

*\*Các kiểu dữ liệu cơ bản trong java script:*

|  |  |  |
| --- | --- | --- |
| [**String**](https://developer.mozilla.org/en-US/docs/Glossary/String) | Một chuỗi các ký tự hay còn gọi là xâu. Để thông báo giá trị nào đó là xâu, bạn nên đặt nó trong dấu nháy. | let myVariable = 'Bob'; hoặc let myVariable = "Bob"; |
| [**Number**](https://developer.mozilla.org/en-US/docs/Glossary/Number) | Số. Không đặt số trong dấu nháy. | let myVariable = 10; |
| [**Boolean**](https://developer.mozilla.org/vi/docs/Tu-dien-thuat-ngu/Boolean) | Giá trị True/False. true và false là các từ khoá đặc biệt trong JS, và không cần tới dấu nháy. | let myVariable = true; |
| [**Array**](https://developer.mozilla.org/vi/docs/Tu-dien-thuat-ngu/array) | Cấu trúc cho phép bạn lưu trữ nhiều giá trị trong một tham chiếu đơn. | let myVariable = [1,'Bob','Steve',10]; Trỏ vào mỗi phần tử của mảng theo cách sau: myVariable[0], myVariable[1], vân vân. |
| [**Object**](https://developer.mozilla.org/en-US/docs/Glossary/Object) | Về cơ bản là bất cứ thứ gì. Mọi thứ trong JavaScript đều là object, và có thể lưu trữ trong biến. Hãy ghi nhớ điều này trong lúc học. | let myVariable = document.querySelector('h1'); Tất cả các ví dụ phía trên cũng vậy. |

**II. Hàm trong JS:**

***1. Cú pháp hàm: (các ví dụ đã đặt trong thẻ <script>)***

- Cú pháp đơn giản của một hàm trong javascript đơn giản như sau:

<**script** type=*"text/javascript"*>

**function** functionname(parameter-list)

{

//statements

}

</**script**>

- Ví dụ:

<**script** type=*"text/javascript"*>

<!--

**function** sayHello()

{

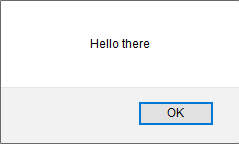
alert("Hello there");

}

//-->

</**script**>

=> Kết quả: khi có một sự kiện nào đó gọi tới hàm này thì hộp thoại như sau sẽ xuất hiện:



- Gọi hàm:

<**html**>

<**head**>

<**script** type=*"text/javascript"*>

**function** sayHello()

{

document.write ("Hello there!");

}

</**script**>

</**head**>

<**body**>

<**p**>Click the following button to call the function</**p**>

<**form**>

<**input** type=*"button"* onclick="sayHello()" value=*"Xin chao!"*>

</**form**>

<**p**>Use different text in write method and then try...</**p**>

</**body**>

</**html**>

- Tham số của hàm:

<**html**>

<**head**>

<**script** type=*"text/javascript"*>

**function** sayHi(name) {

alert("hello " + name);

}

</**script**>

</**head**>

<**body**>

<**p**>Click the following button to call the function</**p**>

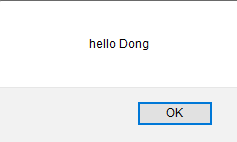
<**a** href=*"#"* onclick="sayHi('Dong');">Click</**a**>

<**p**>Use different parameters inside the function and then try...</**p**>

</**body**>

</**html**>

=> Kết quả: khi bấm vào link:  thì alertbox xuất hiện:

 và biến Dong được truyền vào.

- Hàm lồng trong hàm:

<**html**>

<**head**>

<**script** type=*"text/javascript"*>

**function** concatenate(first, last)

{

**var** full;

full = first + last;

**return** full;

}

**function** secondFunction()

{

**var** result;

result = concatenate('Zara', 'Ali');

document.write (result );

}

</**script**>

</**head**>

<**body**>

<**p**>Click the following button to call the function</**p**>

<**form**>

<**input** type=*"button"* onclick="secondFunction()" value=*"Call Function"*>

<**form**>

<**p**>Use different parameters inside the function and then try...</**p**>

</**body**>

</**html**>

***2. Danh sách các hàm cơ bản trong JS:***

*\*Danh sách các phương thức có sẵn của Đối tượng number:*

|  |  |
| --- | --- |
| [constructor()](https://vietjack.com/javascript/thuoc_tinh_number_constructor_trong_javascript.jsp) | Trả về hàm mà tạo sự instance của đối tượng này. Theo mặc định, nó là đối tượng Number. |
| [toExponential()](https://vietjack.com/javascript/number_toexponential_trong_javascript.jsp) | Làm một số hiển thị ở dạng số mũ, ngay cả khi số này là trong một dãy trong đó JavaScript thường sử dụng ký hiệu chuẩn. |
| [toFixed()](https://vietjack.com/javascript/number_tofixed_trong_javascript.jsp) | Định dạng một số với một số chữ số đặc trưng tới bên phải của phần thập phân. |
| [toLocaleString()](https://vietjack.com/javascript/number_tolocalestring_trong_javascript.jsp) | Trả về một phiên bản giá trị chuỗi của số hiện tại trong một định dạng mà có thể đa dạng theo thiết lập nội bộ của trình duyệt. |
| [toPrecision()](https://vietjack.com/javascript/number_toprecision_trong_javascript.jsp) | Định nghĩa bao nhiêu chữ số (bao gồm các chữ số ở bên phải và bên trái phần thập phân) để hiển thị một số. |
| [toString()](https://vietjack.com/javascript/number_tostring_trong_javascript.jsp) | Trả về sự biểu diễn chuỗi của giá trị số. |
| [valueOf()](https://vietjack.com/javascript/number_valueof_trong_javascript.jsp) | Trả về giá trị số của số. |

*\*Các phương thức của đối tượng Boolean:*

*­*

|  |  |
| --- | --- |
| [toSource()](https://vietjack.com/javascript/boolean_tosource_trong_javascript.jsp) | Trả về một chuỗi chứa nguồn của đối tượng Boolean; bạn có thể sử dụng chuỗi này để tạo một đối tượng tương đương. |
| [toString()](https://vietjack.com/javascript/boolean_tostring_trong_javascript.jsp) | Trả về một chuỗi hoặc "true" hoặc "false" phụ thuộc vào giá trị của đối tượng. |
| [valueOf()](https://vietjack.com/javascript/boolean_valueof_trong_javascript.jsp) | Trả về giá trị gốc của đối tượng Boolean. |

*\*Các phương thức của đối tượng String:*

|  |  |
| --- | --- |
| [charAt()](https://vietjack.com/javascript/string_charat_trong_javascript.jsp) | Trả về ký tự tại chỉ mục (index) đã xác định. |
| [charCodeAt()](https://vietjack.com/javascript/string_charcodeat_trong_javascript.jsp) | Trả về một số chỉ dẫn giá trị Unicode của ký tự tại chỉ mục đã cho |
| [concat()](https://vietjack.com/javascript/string_concat_trong_javascript.jsp) | so khớp văn bản của hai chuỗi và trả về một chuỗi mới. |
| [indexOf()](https://vietjack.com/javascript/indexof_trong_javascript.jsp) | Trả về chỉ mục trong đối tượng String đang gọi của sự xảy ra đầu tiên của giá trị đã xác định, hoặc -1 nếu không tìm thấy. |
| [lastIndexOf()](https://vietjack.com/javascript/string_lastindexof_trong_javascript.jsp) | Trả về chỉ mục trong đối tượng String đang gọi của sự xảy ra cuối cùng của giá trị đã xác định, hoặc -1 nếu không tìm thấy. |
| [localeCompare()](https://vietjack.com/javascript/string_localecompare_trong_javascript.jsp) | Trả về một số chỉ dẫn có hay không một chuỗi tham chiếu đến trước hoặc sau hoặc là giống với chuỗi đã cho trong thứ tự phân loại |
| [length()](https://vietjack.com/javascript/thuoc_tinh_string_length_trong_javascript.jsp) | Trả về độ dài của chuỗi |
| [match()](https://vietjack.com/javascript/string_match_trong_javascript.jsp) | Được sử dụng để so khớp một Regular Expression với một chuỗi. |
| [replace()](https://vietjack.com/javascript/string_replace_trong_javascript.jsp) | Được sử dụng để tìm một so khớp giữa Regular Expression đã cho và một chuỗi, và để thay thế chuỗi phụ đã so khớp với một chuỗi phụ mới. |
| [search()](https://vietjack.com/javascript/string_search_trong_javascript.jsp) | Thực thi việc tìm kiếm cho một match giữa một Regular Expression và một chuỗi đã xác định. |
| [slice()](https://vietjack.com/javascript/string_slice_trong_javascript.jsp) | Trích (extract) một khu vực của một chuỗi và trả về một chuỗi mới. |
| [split()](https://vietjack.com/javascript/string_split_trong_javascript.jsp) | Chia một đối tượng String thành một mảng của các chuỗi bằng việc phân chia chuỗi thành các chuỗi phụ |
| [substr()](https://vietjack.com/javascript/string_substr_trong_javascript.jsp) | Trả về các ký tự trong một chuỗi bắt đầu tại vị trí đã xác định với một số ký tự đã xác định. |
| [substring()](https://vietjack.com/javascript/string_substring_trong_javascript.jsp) | Trả về các ký tự trong một chuỗi giữa hai chỉ mục (index) bên trong chuỗi đó. |
| [toLocaleLowerCase()](https://vietjack.com/javascript/string_tolocalelowercase_trong_javascript.jsp) | Các ký tự bên trong một chuỗi được biến đổi thành kiểu chữ thường đồng thời tôn trọng Locale hiện tại. |
| [toLocaleUpperCase()](https://vietjack.com/javascript/string_tolocaleuppercase_trong_javascript.jsp) | Các ký tự bên trong một chuỗi được biến đổi thành kiểu chữ hoa đồng thời tôn trọng Locale hiện tại. |
| [toLowerCase()](https://vietjack.com/javascript/string_tolowercase_trong_javascript.jsp) | Trả về giá trị chuỗi đang gọi được biến đổi thành kiểu chữ thường |
| [toString()](https://vietjack.com/javascript/string_tostring_trong_javascript.jsp) | Trả về một chuỗi biểu diễn đối tượng đã xác định |
| [toUpperCase()](https://vietjack.com/javascript/string_touppercase_trong_javascript.jsp) | Trả về giá trị chuỗi đang gọi được biến đổi thành kiểu chữ hoa |
| [valueOf()](https://vietjack.com/javascript/string_valueof_trong_javascript.jsp) | Trả về giá trị gốc của đối tượng đã xác định. |

- String HTML Wrapper:

|  |  |
| --- | --- |
| [anchor()](https://vietjack.com/javascript/string_anchor_trong_javascript.jsp) | Tạo một HTML Anchor mà được sử dụng như là một mục tiêu siêu văn bản (hypertext target). |
| [big()](https://vietjack.com/javascript/string_big_trong_javascript.jsp) | Tạo một chuỗi để được hiển thị trong một big font như thể nó trong một thẻ <big> |
| [blink()](https://vietjack.com/javascript/string_blink_trong_javascript.jsp) | Tạo một chuỗi nhấp nháy như thể nó trong một thẻ <blink> |
| [bold()](https://vietjack.com/javascript/string_bold_trong_javascript.jsp) | Tạo một chuỗi để được hiển thị ở dạng in đậm như thể nó trong một thẻ <b> |
| [fixed()](https://vietjack.com/javascript/string_fixed_trong_javascript.jsp) | Tạo một chuỗi để được hiển thị trong một font cố định như thể nó trong một thẻ <tt> |
| [fontcolor()](https://vietjack.com/javascript/string_fontcolor_trong_javascript.jsp) | Tạo một chuỗi để được hiển thị theo một màu cố định như khi nó trong một thẻ <font color="color"> |
| [fontsize()](https://vietjack.com/javascript/string_fontsize_trong_javascript.jsp) | Làm một chuỗi được hiển thị trong một kích cỡ font xác định như khi nó trong một thẻ <font size="size"> |
| [italics()](https://vietjack.com/javascript/string_italics_trong_javascript.jsp) | Làm một chuỗi ở dạng in nghiêng như khi nó trong một thẻ <i> |
| [link()](https://vietjack.com/javascript/string_link_trong_javascript.jsp) | Tạo một HTML link siêu văn bản mà yêu cầu URL khác |
| [small()](https://vietjack.com/javascript/string_small_trong_javascript.jsp) | Làm một chuỗi hiển thị ở dạng Small font như khi nó trong một thẻ <small> |
| [strike()](https://vietjack.com/javascript/string_strike_trong_javascript.jsp) | Làm một chuỗi hiển thị ở dạng văn bản gạch ngang như khi nó trong một thẻ <strike> |
| [sub()](https://vietjack.com/javascript/string_sub_trong_javascript.jsp) | Làm một chuỗi hiển thị ở dạng chỉ số dưới như khi nó trong một thẻ <sub> |
| [sup()](https://vietjack.com/javascript/string_sup_trong_javascript.jsp) | Làm một chuỗi hiển thị ở dạng chỉ số trên như khi nó trong một thẻ <sup> |

- Đối tượng Array:

|  |  |
| --- | --- |
| [concat()](https://vietjack.com/javascript/array_concat_trong_javascript.jsp) | Trả về một mảng mới bao gồm mảng này kết hợp với các mảng khác và/hoặc giá trị khác. |
| [every()](https://vietjack.com/javascript/array_every_trong_javascript.jsp) | Trả về true nếu mỗi phần tử trong mảng này thỏa mãn hàm kiểm tra đã cho. |
| [filter()](https://vietjack.com/javascript/array_filter_trong_javascript.jsp) | Tạo một mảng mới với tất cả các phần tử của mảng này, mà hàm lọc đã cho trả về true. |
| [forEach()](https://vietjack.com/javascript/array_foreach_trong_javascript.jsp) | Gọi một hàm cho mỗi phần tử trong mảng. |
| [indexOf()](https://vietjack.com/javascript/array_indexof_trong_javascrip.jsp) | Trả về chỉ mục đầu tiên (thấp nhất) của một phần tử trong mảng tương đương với giá trị đã cho, hoặc -1 nếu không được tìm thấy. |
| [join()](https://vietjack.com/javascript/array_join_trong_javascript.jsp) | Kết hợp tất cả phần tử trong một mảng vào trong một chuỗi. |
| [lastIndexOf()](https://vietjack.com/javascript/array_lastindexof_trong_javascript.jsp) | Trả về chỉ mục cuối cùng (lớn nhất) của một phần tử trong mảng tương đương với giá trị đã cho, hoặc -1 nếu không được tìm thấy. |
| [map()](https://vietjack.com/javascript/array_map_trong_javascript.jsp) | Tạo một mảng mới với các kết quả của việc gọi một hàm đã cho trên mỗi phần tử của mảng này. |
| [pop()](https://vietjack.com/javascript/array_pop_trong_javascript.jsp) | Gỡ bỏ phần tử cuối cùng từ một mảng và trả về phần tử đó. |
| [push()](https://vietjack.com/javascript/array_push_trong_javascript.jsp) | Thêm một hoặc nhiều phần tử tới phần cuối của một mảng và trả về độ dài mới của mảng. |
| [reduce()](https://vietjack.com/javascript/array_reduce_trong_javascript.jsp) | Áp dụng một hàm đồng thời với hai giá trị của mảng (từ trái qua phải) khi để giảm nó tới một giá trị đơn. |
| [reduceRight()](https://vietjack.com/javascript/array_reduceright_trong_javascript.jsp) | Áp dụng một hàm đồng thời với hai giá trị của mảng (từ phải qua trái) khi để giảm nó tới một giá trị đơn. |
| [reverse()](https://vietjack.com/javascript/array_reverse_trong_javascript.jsp) | Đảo ngược thứ tự của các phần tử của một mảng – Đầu tiên trở thành cuối cùng và cuối cùng trở thành đầu tiên. |
| [shift()](https://vietjack.com/javascript/array_shift_trong_javascript.jsp) | Gỡ bỏ phần tử đầu tiên từ một mảng và trả về phần tử đó. |
| [slice()](https://vietjack.com/javascript/array_slice_trong_javascript.jsp) | Extract – trích một khu vực của một mảng và trả về một mảng mới. |
| [some()](https://vietjack.com/javascript/array_some_trong_javascript.jsp) | Trả về true nếu có ít nhất một phần tử trong mảng này thỏa mãn hàm kiểm tra đã cho. |
| [toSource()](https://vietjack.com/javascript/array_tosource_trong_javascript.jsp) | Biểu diễn code nguồn của một đối tượng. |
| [sort()](https://vietjack.com/javascript/array_sort_trong_javascript.jsp) | Sắp xếp phân loại các phần tử của một mảng. |
| [splice()](https://vietjack.com/javascript/array_splice_trong_javascript.jsp) | Thêm và/hoặc gỡ bỏ các phần tử từ một mảng. |
| [toString()](https://vietjack.com/javascript/array_tostring_trong_javascript.jsp) | Trả về một chuỗi biểu diễn mảng đó và các phần tử của nó. |
| [unshift()](https://vietjack.com/javascript/array_unshift_trong_javascript.jsp) | Thêm một hoặc nhiều phần tử tới phần đầu của một mảng và trả về độ dài mới của mảng. |

- Đối tượng Math:

|  |  |
| --- | --- |
| [abs()](https://vietjack.com/javascript/math_abs_trong_javascript.jsp) | Trả về trị tuyệt đối của một số |
| [acos()](https://vietjack.com/javascript/math_acos_trong_javascript.jsp) | Trả về arccos (giá trị radians) của một số |
| [asin()](https://vietjack.com/javascript/math_asin_trong_javascript.jsp) | Trả về arcsin (giá trị radians) của một số |
| [atan()](https://vietjack.com/javascript/math_atan_trong_javascript.jsp) | Trả về arctan (giá trị radians) của một số |
| [atan2()](https://vietjack.com/javascript/math_atan2_trong_javascript.jsp) | Trả về tang lượng giác ngược |
| [ceil()](https://vietjack.com/javascript/math_ceil_trong_javascript.jsp) | Trả về số integer nhỏ nhất lớn hơn hoặc bằng một số |
| [cos()](https://vietjack.com/javascript/math_cos_trong_javascript.jsp) | Trả về cos của một số. |
| [exp()](https://vietjack.com/javascript/math_exp_trong_javascript.jsp) | Trả về EN, với N là tham số, và E là hằng số Euler |
| [floor()](https://vietjack.com/javascript/math_floor_trong_javascript.jsp) | Trả về số integer lớn nhất nhỏ hơn hoặc bằng một số |
| [log()](https://vietjack.com/javascript/math_log_trong_javascript.jsp) | Trả về logarit cơ số e của một số |
| [max()](https://vietjack.com/javascript/math_max_trong_javascript.jsp) | Trả về số lớn nhất của 0 hoặc nhiều số |
| [min()](https://vietjack.com/javascript/math_min_trong_javascript.jsp) | Trả về số nhỏ nhất của 0 hoặc nhiều số |
| [pow()](https://vietjack.com/javascript/math_pow_trong_javascript.jsp) | Trả về giá trị (cơ số)(số mũ) |
| [random()](https://vietjack.com/javascript/math_random_trong_javascript.jsp) | Trả về một số ngẫu nhiên giữa hai số 0 và 1. |
| [round()](https://vietjack.com/javascript/math_round_trong_javascript.jsp) | Làm tròn số. Trả về giá trị của một số integer gần nhất. |
| [sin()](https://vietjack.com/javascript/math_sin_trong_javascript.jsp) | Trả về sin của một số |
| [sqrt()](https://vietjack.com/javascript/math_sqrt_trong_javascript.jsp) | Trả về căn bậc hai của một số |
| [tan()](https://vietjack.com/javascript/math_tan_trong_javascript.jsp) | Trả về tan của một số |
| [toSource()](https://vietjack.com/javascript/math_tosource_trong_javascript.jsp) | Trả về chuỗi "Math" |

***3. Sự kiện trong JS:***

*\*Kiểu sự kiện onClick: (nhấn chuột trái)*

*\*Kiểu sự kiện onsubmit: là kiểu sự kiện xảy ra khi bạn cố gắng trình một form, nếu hàm validate() trả về true thì form sẽ được đệ trình còn không thì form không được đệ trình dữ liệu. (cái này phải thực hành thật thì mới biết được, bên dưới phần này sẽ extend).*

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Insert title here</**title**>

<**script** type=*"text/javascript"*>

<!--

function validation() {

all validation goes here

.........

return either true or false

}

//-->

</**script**>

</**head**>

<**body**>

<**form** method=*"POST"* action=*"response"* onsubmit="return validate();">

....... <**input** type=*"submit"* value=*"Submit"* />

</**form**>

</**body**>

</**html**>

*\*Kiểu sự kiện onmouseover và onmouseout: hai kiểu sự kiện này sẽ tạo các hiệu quả đẹp với văn bản và hình ảnh. Sự kiện onmouseover kích hoạt khi bạn di chuyển chuột qua bất kỳ phần tử nào và onmouseout kích hoạt khi chuột di chuyển ra khỏi khu vực đó.*

*onmouseout:*

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Insert title here</**title**>

<**script** type=*"text/javascript"*>

<!--

function over() {

document.write("Mouse Over");

}

function out() {

document.write("Mouse Out");

}

//-->

</**script**>

</**head**>

<**body**>

<**p**>Bring your mouse inside the division to see the result:</**p**>

<**div** onmouseout="out()">

<**h2**>This is inside the division</**h2**>

</**div**>

</**body**>

</**html**>

*­ onmouseover:*

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Insert title here</**title**>

<**script** type=*"text/javascript"*>

<!--

function over() {

document.write("Mouse Over");

}

function out() {

document.write("Mouse Out");

}

//-->

</**script**>

</**head**>

<**body**>

<**p**>Bring your mouse inside the division to see the result:</**p**>

<**div** onmouseover="over()">

<**h2**>This is inside the division</**h2**>

</**div**>

</**body**>

</**html**>

*\*Các sự kiện HTML5 chuẩn:*

|  |  |  |
| --- | --- | --- |
| Offline | script | Kích hoạt khi tài liệu ở ngoại tuyến |
| Onabort | script | Kích hoạt trên một sự kiện bỏ dở |
| onafterprint | script | Kích hoạt sau khi tài liệu được in |
| onbeforeonload | script | Kích hoạt trước khi tài liệu tải |
| onbeforeprint | script | Kích hoạt trước khi tài liệu được in |
| onblur | script | Kích hoạt khi cửa sổ mất trọng tâm |
| oncanplay | script | Kích hoạt khi media có thể bắt đầu chơi, nhưng phải dừng để đệm |
| oncanplaythrough | script | Kích hoạt khi media có thể chơi tới cuối, mà không dừng để đệm |
| onchange | script | Kích hoạt khi một phần tử thay đổi |
| onclick | script | Kích hoạt trên một cú click chuột |
| oncontextmenu | script | Kích hoạt khi menu ngữ cảnh bị kích hoạt |
| ondblclick | script | Kích hoạt khi nhấp đúp chuột |
| ondrag | script | Kích hoạt khi một phần tử bị kéo |
| ondragend | script | Kích hoạt tại phần cuối của hoạt động kéo |
| ondragenter | script | Kích hoạt khi một phần tử đã được kéo tới một mục tiêu thả hợp lệ |
| ondragleave | script | Kích hoạt khi một phần tử đang được kéo qua một mục tiêu thả hợp lệ |
| ondragover | script | Kích hoạt tại phần đầu của hoạt động kéo |
| ondragstart | script | Kích hoạt tại phần đầu của hoạt động kéo |
| ondrop | script | Kích hoạt khi phần tử được kéo đang được thả |
| ondurationchange | script | Kích hoạt khi độ dài của media được thay đổi |
| onemptied | script | Kích hoạt khi phần tử nguồn media đột nhiên trở nên trống |
| onended | script | Kích hoạt khi media tiến tới cuối cùng |
| onerror | script | Kích hoạt khi một lỗi xảy ra |
| onfocus | script | Kích hoạt khi cửa sổ nhận trọng tâm |
| onformchange | script | Kích hoạt khi một form thay đổi |
| onforminput | script | Kích hoạt khi một form nhận input từ người dùng |
| onhaschange | script | Kích hoạt khi tài liệu có thay đổi |
| oninput | script | Kích hoạt khi một phần tử nhận đầu vào từ người dùng |
| oninvalid | script | Kích hoạt khi một phần tử không hợp lệ |
| onkeydown | script | Kích hoạt khi một phím bị nhấn |
| onkeypress | script | Kích hoạt khi một phím bị nhấn và được thả ra |
| onkeyup | script | Kích hoạt khi một phím được thả ra |
| onload | script | Kích hoạt khi tài liệu tải |
| onloadeddata | script | Kích hoạt khi dữ liệu media được tải |
| onloadedmetadata | script | Kích hoạt khi thời gian và dữ liệu khác của một phần tử media được tải |
| onloadstart | script | Kích hoạt khi trình duyệt bắt đầu tải dữ liệu media |
| onmessage | script | Kích hoạt khi thông báo được kích hoạt |
| onmousedown | script | Kích hoạt khi một nút chuột được nhả ra |
| onmousemove | script | Kích hoạt khi con trỏ chuột di chuyển |
| onmouseout | script | Kích hoạt khi con trỏ chuột di chuyển ra khỏi một phần tử |
| onmouseover | script | Kích hoạt khi con trỏ chuột di chuyển qua một phần tử |
| onmouseup | script | Kích hoạt khi một nút chuột được nhả ra |
| onmousewheel | script | Kích hoạt khi bánh xe chuột được quay |
| onoffline | script | Kích hoạt khi tài liệu ở ngoại tuyến |
| onoine | script | Kích hoạt khi tài liệu ở trực tuyến |
| ononline | script | Kích hoạt khi tài liệu ở trực tuyến |
| onpagehide | script | Kích hoạt khi cửa sổ bị ẩn |
| onpageshow | script | Kích hoạt khi cửa sổ trở nên nhìn thấy |
| onpause | script | Kích hoạt khi dữ liệu media bị dừng |
| onplay | script | Kích hoạt khi dữ liệu media chuẩn bị bắt đầu chơi |
| onplaying | script | Kích hoạt khi dữ liệu media đang bắt đầu chơi |
| onpopstate | script | Kích hoạt khi lịch sử cửa sổ thay đổi |
| onprogress | script | Kích hoạt khi trình duyệt đang nhận dữ liệu media |
| onratechange | script | Kích hoạt khi tốc độ chơi của media đã thay đổi |
| onreadystatechange | script | Kích hoạt khi trạng thái sẵn sàng thay đổi |
| onredo | script | Kích hoạt khi tài liệu thực hiện một redo |
| onresize | script | Kích hoạt khi cửa sổ bị thay đổi kích cỡ |
| onscroll | script | Kích hoạt khi thanh cuốn của phần tử đang được cuốn |
| onseeked | script | Kích hoạt khi thuộc tính seeking của một phần tử media không còn true, và seeking đã kết thúc |
| onseeking | script | Kích hoạt khi thuộc tính seeking của một phần tử media là true, và seeking đã bắt đầu |
| onselect | script | Kích hoạt khi một phần tử được chọn |
| onstalled | script | Kích hoạt khi có một lỗi trong quá trình thu nhận dữ liệu media |
| onstorage | script | Kích hoạt khi một tài liệu tải |
| onsubmit | script | Kích hoạt khi một form được đệ trình |
| onsuspend | script | Kích hoạt khi trình duyệt đã đang thu nhận dữ liệu media, nhưng đã dừng lại trước khi toàn bộ media file được thu nhận |
| ontimeupdate | script | Kích hoạt khi media thay đổi vị trí chơi của nó |
| onundo | script | Kích hoạt khi một tài liệu thực hiện một undo |
| onunload | script | Kích hoạt khi người sử dụng rời khỏi tài liệu |
| onvolumechange | script | Kích hoạt khi media thay đổi âm lượng, kể cả khi media được thiết lập là mute |
| onwaiting | script | Kích hoạt khi media đã dừng chơi nhưng được mong chờ phục hồi. |

***4. Page Redirect:***

- Đôi khi bấm vào url một trang X nhưng lại được chuyển tới trang Y, khi đó gọi là page redirect (điều hướng lại trang).

- Sử dụng khi: bạn muốn trang web của bạn điều hướng qua một tên miền khác mà bạn thích hơn, hoặc xây các trang khác nhau cho các trình duyệt khác nhau nhưg không muốn điều hướng tại serverside mà tị client. Hoặc Search Engines có thể đã Index các trang của bạn. Nhưng trong khi di chuyển tới miền mới, bạn không muốn đánh mất những vị khách truy cập thông qua phương tiện tìm kiếm. Vì thế bạn có thể sử dụng sự điều hướng lại trang ở Client-Side. Nhưng bạn nên nhớ rằng điều này không nên được thực hiện để lừa dối Search Engine, nó có thể cho trang của bạn thành trang bị cấm.

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Insert title here</**title**>

<**script** type=*"text/javascript"*>

<!--

function Redirect() {

window.location = "https://www.facebook.com/";

}

//-->

</**script**>

</**head**>

<**body**>

<**p**>Click the following button, you will be redirected to home page.</**p**>

<**form**>

<**input** type=*"button"* value=*"Redirect Me"* onclick="Redirect();" />

</**form**>

</**body**>

</**html**>

- Có thể hiển thị một thông báo cho khách hàng trước khi thực hiện chuyển qua một trang mới. Với hàm setTimeOut có sẵn trong JS, là một hàm cho pháp một thực thi một hàm khác sau một khoảng thời gian.

setTimeout('Redirect()', 10000);

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Insert title here</**title**>

<**script** type=*"text/javascript"*>

<!--

function Redirect() {

window.location = "https://www.facebook.com/";

}

document.write("You will be redirected to main page in 10 sec.");

setTimeout('Redirect()', 10000);

//-->

</**script**>

</**head**>

<**body**>

</**body**>

</**html**>

- Điều hướng dựa trên trình duyệt đang sử dụng:

<**html**>

<**head**>

<**script** type=*"text/javascript"*>

<!--

var browsername = navigator.appName;

if (browsername == "Netscape") {

window.location = "http://www.location.com/ns.jsp";

} else if (browsername == "Microsoft Internet Explorer") {

window.location = "http://www.location.com/ie.jsp";

} else {

window.location = "http://www.location.com/other.jsp";

}

//-->

</**script**>

</**head**>

<**body**>

</**body**>

</**html**>

***5. Hộp thoại:***

- Trong JS có 3 kiểu hộp thoại quan trọng mặc định trong JS: AlertBox, Confirmation Box, Prompt Dialog Box.

*\*Alert Dialog Box: Là hộp thoại thông báo phổ biến nhất, thường được dùng để debug. Đây là một hàm của đối tượng window (đối tượng cửa sổ trình duyệt).*

<**html**>

<**head**>

<**script** type=*"text/javascript"*>

<!--

function Warn() {

alert ("This is a warning message!");

document.write ("This is a warning message!");

}

//-->

</**script**>

</**head**>

<**body**>

<**p**>Click the following button to see the result: </**p**>

<**form**>

<**input** type=*"button"* value=*"Click Me"* onclick="Warn();" />

</**form**>

</**body**>

</**html**>

*\*Comfirmation Dialog Box: Dùng để xác nhận sự đồng ý của người dùng trên các tuỳ chọn, hiển thị thông qua phương thức* ***confirm()****. Bấm OK thì phương thức* ***confirm()*** *trả về true còn không thì trả về false.*

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Insert title here</**title**>

<**script** type=*"text/javascript"*>

<!--

function getConfirmation() {

var retVal = confirm("Do you want to continue ?");

if (retVal == true) {

document.write("User wants to continue!");

return true;

} else {

Document.write("User does not want to continue!");

return false;

}

}

//-->

</**script**>

</**head**>

<**body**>

<**p**>Click the following button to see the result:</**p**>

<**form**>

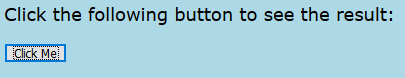
<**input** type=*"button"* value=*"Click Me"* onclick="getConfirmation();" />

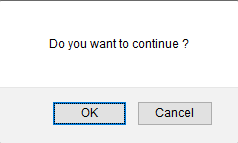
</**form**>

</**body**>

</**html**>

=> Kết quả:





*\*Prompt Dialog Box: Hộp thoại này rất hữu ích khi muốn popup một hộp văn bản để nhận input từ người dùng, hiển thị thông qua phương thức* ***prompt()****. Nếu người dùng bấm OK thì phương thức này trả về giá trị nhập vào từ hộp văn bản, nếu bấm Cancel thì phương thức* ***prompt()*** *trả về null.*

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Insert title here</**title**>

</**script**>

<**script** type=*"text/javascript"*>

<!--

function getValue(){

var retVal = prompt("Enter your name : ", "your name here");

document.write("You have entered : " + retVal);

}

//-->

</**script**>

</**head**>

<**body**>

<**p**>Click the following button to see the result: </**p**>

<**form**>

<**input** type=*"button"* value=*"Click Me"* onclick="getValue();" />

</**form**>

</**body**>

</**html**>

***5. Từ khoá void:***

- void là một từ khoá quan trọng trong JS: có thể được sử dụng như là một toán tử một ngôi xuất hiện trước toán hạng đơn của nó, toán từ này không trả về bất kỳ một giá trị nào ☺

<**head**>

<**script** type=*"text/javascript"*>

<!--

void func()

javascript:void func()

or:

void(func())

javascript:void(func())

//-->

</**script**>

</**head**>

*\*Ứng dụng của void:*

*­*- Toán tử này sử dụng phổ biến trong Client-side: nó cho phép ước lượng một biểu thức cho các tác động thứ yếu của nó, mà trình duyệt không trả về giá trị của nó.

<a href="javascript:void(alert('Warning!!!'))">Click me!</a>

- Tạo ra các giá trị undifine.

**III. Hướng đối tượng trong JS:**

*\*JS là một ngôn ngữ lập trình hướng đối tượng với 4 tính chất cơ bản: trừu tượng, đóng gói, kế thừa, đa hình.*

- Các thuộc tính của đối tượng: cú pháp để thêm một thuộc tính tới đối tượng là:

objectName.objectProperty = propertyValue;

- Phương thức đối tượng:

+ Có thể tham chiếu bằng từ khoá this.

+ Gọi thông qua toán tử “.’”: document.write("This is test");

- Đối tượng người dùng tự định nghĩa:

+ Dùng toán tử new để tạo ra các đối tượng mới.

var employee = new Object();

var books = new Array("C++", "Perl", "Java");

var day = new Date("August 15, 1947");

+ JS cung cấp một constructor khá đặc biệt đó là **Object()** giá trị trả về của constructor này được gán tới một biến và biến này tham chiều tới một đối tượng mới. Các thuộc tính được gán tới đối tượng không là biến và không được định nghĩa bởi từ khoá **var.**

<**html**>

<**head**>

<**title**>User-defined objects</**title**>

<**script** type=*"text/javascript"*>

**var** book = **new** Object(); // Create the object

book.subject = "Perl"; // Assign properties to the object

book.author = "Mohtashim";

</**script**>

</**head**>

<**body**>

<**script** type=*"text/javascript"*>

document.write("Book name is : " + book.subject + "<br>");

document.write("Book author is : " + book.author + "<br>");

</**script**>

</**body**>

</**html**>

+ Khai báo bằng từ khoá function:

<**script** type=*"text/javascript"*>

**function** myValidate1() {

}

**var** myObj = **new** myValidate1();

document.write(**typeof** myObj);

</**script**>

=> Kiểu tra về là object.

+ Tạo đối tượng bằng từ khoá this:

**this**.myValidate2 = **function**() {

}

**var** myObj = **new** myValidate2();

document.write(**typeof** myObj);

**var** my = **new**

+ Tạo bằng từ khoá var:

**var** myObj3 = {}

+ Tạo một đối tượng hoàn chỉnh (có thuộc tính và phương thức):

**var** enter = "<br/>";

**this**.myValidate = **function**() {

**this**.data = "Data";

// Hien thuc phuong thuc.

**this**.getData = **function**() {

**return** **this**.data;

}

} + Truy cập thuộc tính và phương thức:

**var** enter = "<br/>";

**this**.myValidate = **function**() {

**this**.data = "Data";

// Hien thuc phuong thuc.

**this**.getData = **function**() {

**return** **this**.data;

}

}

**var** myObj = **new** myValidate();

document.write(myObj.data + enter);

document.write(myObj.getData());

- public và private trong JS:

**var** enter = "<br/>";

**this**.myValidate = **function**() {

**this**.data = "Data"; // Thuộc tính public

**var** private\_data = "Private data"; // Thuộc tính private

// Hien thuc phuong thuc.

**this**.getData = **function**() {

**return** **this**.data;

}

}

**var** myObj = **new** myValidate();

document.write(myObj.data+enter);

document.write(myObj.private\_data);

=> Kết quả:



+ Thuộc tính hay phương thức private thì chỉ có thể sử dụng được trong phạm vi khai báo của đối tượng. Để tạo ra một thuộc tính hay phương thức private, sử dụng từ khoá var + tên thuộc tính/phương thức.

**var** private\_data = "Private data";

Lấy thuộc tính hay phương thức private như sau:

**var** enter = "<br/>";

**this**.myValidate = **function**() {

**var** private\_data = "Private data";

**this**.getPrivateData = **function**() {

**return** private\_data;

}

}

**var** myObj = **new** myValidate();

document.write(myObj.getPrivateData());

=> Kết quả: 

+ public thì có thể truy cập từ ngoài phạm vi khai báo của đối tượng.

- Truyền tham số cho đối tượng:

**var** enter = "<br/>";

**this**.myValidate = **function**(value\_1, value\_2) {

**this**.data = value\_1;

**var** private\_data = value\_2;

// Hien thuc phuong thuc.

**this**.getData = **function**() {

**return** **this**.data;

}

**this**.getPrivateData = **function**() {

**return** private\_data;

}

}

**var** myObj = **new** myValidate("Hi", ", Dong!");

document.write(myObj.data);

document.write(myObj.getPrivateData());

=> Kết quả: 

- Truyền tham số cho phương thức:

**var** enter = "<br/>";

**this**.myValidate = **function**(value\_1, value\_2) {

**this**.data = value\_1;

**var** private\_data = value\_2;

// Hien thuc phuong thuc.

**this**.setData = **function**(new\_value)

{

**this**.data = new\_value;

}

**this**.getData = **function**() {

**return** **this**.data;

}

**this**.getPrivateData = **function**() {

**return** private\_data;

}

}

**var** myObj = **new** myValidate("Hi", ", Dong!");

myObj.setData("Hello");

document.write(myObj.data);

document.write(myObj.getPrivateData());

*\*Kế thừa trong JS:*

- Có thể thực hiện kế thừa các thuộc tính và phương thức từ một thuộc tính và phương thức khác bằng từ khoá call:

**var** enter = "<br/>";

**this**.myValidate = **function**(value\_1, value\_2) {

**this**.data = value\_1;

**var** private\_data = value\_2;

// Hien thuc phuong thuc.

**this**.setData = **function**(new\_value) {

**this**.data = new\_value;

}

**this**.getData = **function**() {

**return** **this**.data;

}

**this**.getPrivateData = **function**() {

**return** private\_data;

}

}

**this**.myValidateChild = **function**(value\_1, value\_2) {

myValidate.call(**this**);

}

**var** myObj = **new** myValidateChild("Hi", ", Dong!");

myObj.setData("Hello");

document.write(myObj.data);

document.write(myObj.getPrivateData());

=> Kết quả: 

**IV.**