

JavaScript

Nội dung

- Xử lý sự kiện trong JavaScript
- Mô hình HTML DOM
- Ví dụ minh họa
- Biểu thức quy tắc (Regular expression)

Xử lý sự kiện trong JS

- Sự kiện trong JS
 - Hành động được phát hiện bởi JS
 - Mỗi trang web sẽ có các sự kiện và sự kiện này có thể chặn để xử lý theo ý đồ của người lập trình!
- VD:
 - Sự kiện **onclick** để bắt hành động kích chuột vào một button hay thành phần nào đó.
 - Để định nghĩa hành động gì thực hiện khi sự kiện này diễn ra thì có thể dùng đoạn mã JS hay gọi một hàm nào đó để xử lý cho hành động này.

Xử lý sự kiện trong JS

- Cú pháp

`<tagName eventHandler = "JavaScript-Code or Function">`

- VD: để kiểm tra khi có bất cứ sự thay đổi trên giá trị nhập liệu, ta có thể dùng sự kiện **onchange()** khai báo tới hàm xử lý.

`<input type="text" name="age" onchange="checkage()">`



Hàm xử lý sự kiện onchange

Xử lý sự kiện trong JS

- Các sự kiện trong JS

Thuộc tính	Sự kiện xảy ra khi
onabort	Tải một hình ảnh bị ngắt
onblur	Khi một thành phần bị mất focus
onchange	Người sử dụng thay đổi nội dung trên một field
onclick	Click chuột vào một đối tượng
ondblclick	Double click chuột vào một đối tượng
onerror	Một lỗi xảy ra khi tải một tài liệu hay hình ảnh
onfocus	Một thành phần có focus
onkeydown	Khi một phím trên bàn phím được nhấn xuống
onkeypress	Khi một phím trên bàn phím được nhấn
onkeyup	Khi một phím trên bàn phím được thả ra

Xử lý sự kiện trong JS

- Các sự kiện trong JS

Thuộc tính	Sự kiện xảy ra khi
onload	Một trang hay hình ảnh tải hoàn tất
onmousedown	Khi nút chuột được nhấn
onmousemove	Khi chuột di chuyển
onmouseout	Khi chuột di chuyển ra ngoài một thành phần
onmouseover	Khi chuột di chuyển bên trên một thành phần
onmouseup	Khi nút chuột được thả
onreset	Khi nút Reset được nhấn
onresize	Khi một cửa sổ hay frame thay đổi kích thước
onselect	Khi văn bản được chọn
onsubmit	Khi nhấn nút submit
onunload	Người sử dụng thoát khỏi trang web

Xử lý sự kiện trong JS

- Các sự kiện thường dùng của một số đối tượng

Đối tượng	Các xử lý sự kiện của đối tượng
Selection list	onBlur, onChange, onFocus
Text	onBlur, onChange, onFocus, onSelect
Textarea	onBlur, onChange, onFocus, onSelect
Button	onClick
Checkbox	onClick
Radio button	onClick
Hypertext link	onClick, onMouseOver, onMouseOut
Clickable Imagemap area	onMouseOver, onMouseOut
Reset button	onClick

Xử lý sự kiện trong JS

- Sự kiện của một số đối tượng thông dụng

Đối tượng	Các xử lý sự kiện của đối tượng
Submit button	onClick
Document	onLoad, onUnload, onError
Window	onLoad, onUnload, onBlur, onFocus
Framesets	onBlur, onFocus
Form	onSubmit, onReset
Image	onLoad, onError, onAbort

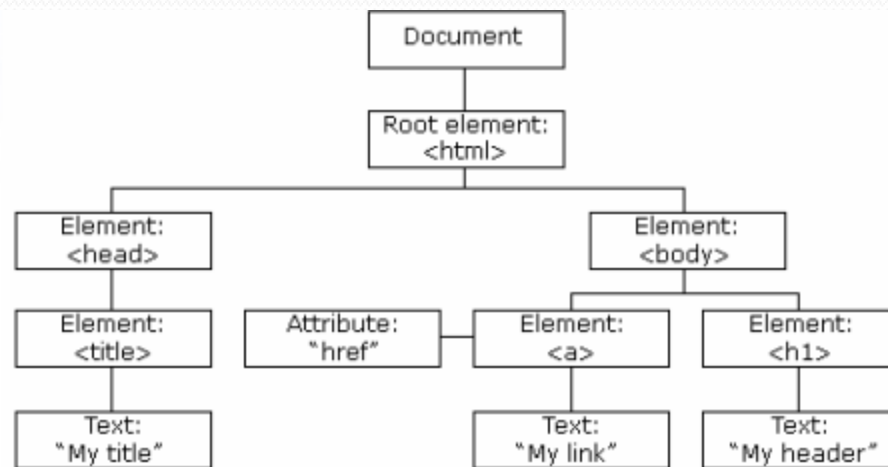
Xử lý sự kiện trong JS

- Chỉ cho phép nhận ký tự trong textbox

```
<script type="text/javascript">  
function noNumbers(e) {  
    var keynum; var keychar; var numcheck;  
    if(window.event) keynum = e.keyCode; // IE  
    else if(e.which) keynum = e.which; // Netscape/Firefox/Opera  
    keychar = String.fromCharCode(keynum);  
    numcheck = /\d/; //regular expression finds any non digit character  
    return !numcheck.test(keychar);  
}  
</script>  
<form><input type="text" onkeydown="return noNumbers(event)" />  
</form>
```

HTML Document Object Model (DOM)

- **HTML DOM**: mô hình đối tượng tài liệu HTML
 - Định nghĩa một chuẩn để truy cập và thao tác trên các tài liệu HTML
- DOM biểu diễn một tài liệu HTML bằng một **cấu trúc cây** (node tree), với các phần tử, thuộc tính và văn bản



HTML Document Object Model (DOM)

- DOM là gì
 - Với JS có thể **tái cấu trúc** lại toàn bộ tài liệu HTML. Có thể thêm, bớt, thay đổi hay sắp xếp lại các phần tử của trang.
 - Để thay đổi mọi thứ trong trang, JS phải truy cập được tất cả các thành phần HTML trong tài liệu. Thông qua DOM, JS có thể truy cập và sửa đổi đến tất cả thành phần của trang .
 - DOM được công bố 1998 và cho đến nay tất cả trình duyệt thông dụng đều tích hợp và hỗ trợ mô hình này.

HTML Document Object Model (DOM)

- DOM là gì
 - Với mô hình DOM, bạn có thể sử dụng JS để đọc và thay đổi các tài liệu như HTML, XHTML và XML.
 - Mô hình DOM chia làm 3 phần
 - **Core DOM**: định nghĩa một tập tài liệu chuẩn cho mọi tài liệu có cấu trúc
 - **XML DOM**: định nghĩa một tập đối tượng chuẩn cho tài liệu XML
 - **HTML DOM**: định nghĩa một tập đối tượng chuẩn cho tài liệu HTML.

HTML Document Object Model (DOM)

- HTML DOM nodes
 - Theo mô hình DOM, mọi thứ trong tài liệu HTML là một nút.
 - Mỗi thẻ HTML là một nút thành phần (element node)
 - Các văn bản chứa trong các thành phần HTML gọi là các nút văn bản (text node)
 - Mỗi thuộc tính của thành phần HTML là một nút thuộc tính (attribute node)
 - Các ghi chú là các node ghi chú (comment node)

HTML Document Object Model (DOM)

- Hệ thống phân cấp của các node
 - Các node có mối quan hệ với node khác
 - Các node trong tài liệu HTML được biểu diễn dưới dạng cây tài liệu (document tree).
 - Document tree bắt đầu tại document node và tiếp tục phân nhánh cho đến khi đến tất cả các text node ở mức thấp nhất của cây.

HTML Document Object Model (DOM)

- HTML DOM node information
 - Mỗi nút có 3 thuộc tính `nodeName`, `nodeValue`, `nodeType` lưu thông tin về nút
 - `nodeName`: chứa tên của nút
 - Với element node, nodeName là tên thẻ của nút
 - Với attribute node, nodeName là tên thuộc tính của nút
 - Với text node, nodeName luôn có tên là #text
 - Với document node, nodeName luôn có tên là #document
 - `nodeValue`:
 - Với text node, nodeValue là nội dung văn bản của nút
 - Với attribute node, nodeValue là giá trị thuộc tính

HTML Document Object Model (DOM)

- HTML DOM node information
 - `nodeType`: lưu thông tin về loại của nút
 - Một số loại thành phần và giá trị tương ứng của `nodeType`

Element type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

HTML Document Object Model (DOM)

- Tất cả trang web đều có đối tượng sau
 - **Window**: đối tượng ở mức cao nhất, có các thuộc tính thực hiện áp dụng trên toàn cửa sổ
 - **Navigator**: đối tượng lưu các thông tin về trình duyệt của client
 - **Screen**: đối tượng lưu các thông tin về màn hình client
 - **History**: đối tượng lưu các URL đã viếng thăm của cửa sổ trình duyệt
 - **Location**: đối tượng lưu thông tin về URL hiện hành
 - **Document**: đối tượng mô tả toàn bộ cấu trúc HTML của tài liệu, có thể sử dụng đối tượng này để truy cập các thành phần trong trang.

HTML Document Object Model (DOM)

- Các thuộc tính của đối tượng Window

Thuộc tính	Mô tả
defaultStatus	Thông báo mặc định của thanh trạng thái
frames[]	Mảng xác định tất cả các frame trong cửa sổ
length	Số lượng các frame trong cửa sổ cha mẹ
name	Tên của cửa sổ hiện thời
parent	Đối tượng cửa sổ cha mẹ
self	Cửa sổ hiện thời
status	Sử dụng để lấy hay đặt lại thông báo trạng thái và ghi đè lên defaultStatus
top	Cửa sổ ở trên cùng
window	Cửa sổ hiện thời

HTML Document Object Model (DOM)

- Các phương thức của đối tượng Window

Phương thức	Mô tả
alert("message")	Hiển thị hộp thoại thông báo với một thông điệp và nút OK
blur()	Bỏ focus tại cửa sổ hiện hành (chuyển focus sang cửa sổ hay đối tượng khác)
clearInterval(interval)	Huỷ bỏ một thiết lập interval do setInterval() thiết lập
clearTimeout(timeout)	Huỷ bỏ một thiết lập timeout do setTimeout() thiết lập
close()	Đóng cửa sổ hiện tại
confirm("message")	Hiển thị một hộp thoại với một thông điệp và một nút OK và nút Cancel
createPopup()	Tạo một cửa pop-up
focus()	Thiết lập focus cho cửa sổ hiện tại
moveBy(x, y)	Di chuyển vị trí của cửa sổ một khoảng so với vị trí hiện tại

HTML Document Object Model (DOM)

- Các phương thức của đối tượng Window

Phương thức	Mô tả
moveTo(x, y)	Di chuyển cửa sổ đến vị trí xác định
open(URL)	Mở một cửa sổ trình duyệt mới
print()	Xuất nội dung của cửa sổ hiện tại
prompt("text")	Hiển thị một hộp thoại yêu cầu user nhập dữ liệu
resizeBy(width,height)	Thay đổi kích thước cửa sổ theo số pixel xác định so với kích thước cửa sổ hiện tại
resizeTo(width,height)	Thay đổi kích thước cửa sổ theo chiều rộng và chiều cao xác định
scrollBy(xnum,ynum)	Cuộn nội dung bởi một số pixel xác định
scrollTo(xpos,ypos)	Cuộn nội dung đến một vị trí xác định
setInterval(interval)	Ước lượng một biểu thức sau một khoảng interval xác định
setTimeout(timeout)	Ước lượng một biểu thức sau một khoảng millisecond xác định

HTML Document Object Model (DOM)

- Các tập hợp mảng của đối tượng document

Tập hợp	Mô tả
anchors[]	Trả về mảng các đối tượng Anchor trong tài liệu
forms[]	Trả về mảng các đối tượng Form trong tài liệu
images[]	Trả về mảng các đối tượng Image trong tài liệu
links[]	Trả về mảng các đối tượng Area và Link trong tài liệu

- Chú ý:
 - Xem lại các thuộc tính và phương thức của mảng để sử dụng tập hợp này
 - Mỗi phần tử của tập hợp này có đầy đủ các thuộc tính và các thành phần HTML tương ứng.

HTML Document Object Model (DOM)

- Các thuộc tính của đối tượng document

Thuộc tính	Mô tả
body	Truy cập trực tiếp vào thẻ <body>
cookie	Thiết lập hay trả về các cookie được kết hợp với tài liệu hiện tại
domain	Trả về tên miền của tài liệu hiện tại
lastModified	Trả về thời điểm cập nhật cuối cùng của tài liệu
referrer	Trả về URL của tài liệu đã tải tài liệu hiện tại
title	Trả về title của tài liệu hiện tại
URL	Trả về URL hiện tại của tài liệu

HTML Document Object Model (DOM)

- Các phương thức của đối tượng document

Phương thức	Mô tả
close()	Đóng luồng xuất được mở với phương thức document.open(), và hiển thị dữ liệu được tập hợp
getElementById(id)	Trả về một tham chiếu của đối tượng đầu tiên với id xác định
getElementsByName(name)	Trả về một tập hợp các đối tượng với tên xác định
getElementsByTagName(tag Name)	Trả về một tập hợp các đối tượng với tên thẻ (tagname) xác định
open(mimetype,replace)	Mở một luồng để tập hợp kết quả xuất từ phương thức document.write() và document.writeln()
write(exp1,exp2,exp3,...) writeln(exp1,exp2,exp3,...)	Xuất các biểu thức HTML hay mã JavaScript vào tài liệu

HTML Document Object Model (DOM)

- VD 1: cách sử dụng hàm setInterval và clearInterval để gọi hàm clock() sau 50 ms


```
<html>
<body>
<input type="text" id="clock" size="35" />
  <script type="text/javascript">
    var int=self.setInterval("clock()",50);
    function clock()
    {
      var t=new Date();
      document.getElementById("clock").value=t;
    }
  </script>
<button onclick="int=window.clearInterval(int)">Stop interval</button>
</body>
</html>
```


HTML Document Object Model (DOM)

- VD2: cách sử dụng hàm setTimeout() và clearTimeout()

```
<html>
<head>
<script type="text/javascript">
  var c=0;
  var t;
  function timedCount(){
    document.getElementById('txt').value=c;
    c=c+1;
    t=setTimeout("timedCount()",1000);
  }
  function stopCount(){
    clearTimeout(t);
  }
</script>
</head>
```

```
<body>
  <form>
    <input type="button" value="Start
      count!" onClick="timedCount()">
    <input type="text" id="txt">
    <input type="button" value="Stop
      count!" onClick="stopCount()">
  </form>
</body>
</html>
```



Click on the "Start count!" button above to start the timer. The input field will count forever, starting at 0. Click on the "Stop count!" button to stop the counting.

Mô hình hướng đối tượng

- JS không phải là NN LT HDT dựa trên lớp như C++, C#, Java.
- JS là dạng HDT dựa trên prototype.
- Trong HDT dựa trên lớp, hai khái niệm lớp và đối tượng được phân biệt rõ ràng. Lớp là mô hình trừu tượng, mô tả tính chất chung cho các đối tượng, trong khi các đối tượng chỉ là các thể hiện cụ thể của lớp nào đó.

Mô hình hướng đối tượng

- Hướng đối tượng dựa trên prototype
 - Chỉ có khái niệm **đối tượng** (không có khái niệm lớp)
 - Định nghĩa một khái niệm mới đó là **đối tượng nguyên mẫu**, đối tượng này được sử dụng như một khuôn mẫu để khởi tạo các đối tượng mới.
 - Các đối tượng có thể thêm bớt thuộc tính và phương thức của chính nó, tại thời điểm tạo đối tượng hay run time.
 - Các đối tượng có thể được kết hợp như là prototype của đối tượng khác để chia sẻ các thuộc tính của nó với các đối tượng khác.

Mô hình hướng đối tượng

- VD:
 - Nếu đối tượng A được kết hợp như là một prototype của đối tượng B thì đối tượng B, ngoài tính chất riêng của nó, sẽ có thêm các tính chất của đối tượng A.
- Khai báo và khởi tạo đối tượng
 - Định nghĩa lớp rỗng

```
function MyClass() { }
```
 - Tạo thể hiện của lớp MyClass

```
var myClassObj = new MyClass();
```

Mô hình hướng đối tượng

- Lưu ý:
 - Nếu thiếu từ khóa new, lệnh sau xem như câu lệnh gọi hàm MyClass, với kết quả trả về được lưu trong biến
`var result = MyClass();`
 - Nếu thiếu từ khóa new và cặp dấu ngoặc (), xem như tạo
`var myfunc = MyClass;`
CÓ THỂ GỌI HÀM THAY THẾ CHO MyClass()

Lúc này để gọi hàm MyClass, ta có thể gọi hàm myfunc

Mô hình đối tượng

- Thêm các thuộc tính động (dynamic property) cho đối tượng

```
myClassObj.myData = 5;  
myClassObj.myString = "Hello World";  
alert( myClassObj.myData );    // displays: 5  
alert( myClassObj.myString );  // displays: "Hello World"
```

- Lưu ý: các thuộc tính động này chỉ tồn tại trong đối tượng được gán động, các đối tượng khác sẽ không có thuộc tính này.

```
var myNewClassObj = new MyClass();  
alert( myNewClassObj.myData );    // displays: undefined
```

Mô hình đối tượng

- Để tạo lớp có các thuộc tính sẽ tồn tại trong tất cả thể hiện (instance) của lớp \Rightarrow dùng từ khóa **this** khai báo các thuộc tính bên trong khai báo lớp.

```
function MyClass()
```

```
{
```

```
  this.myData = 5;
```

```
  this.myString = "Hello World";
```

```
}
```

```
var myClassObj1 = new MyClass();
```

```
var myClassObj2 = new MyClass();
```

```
myClassObj1.myData = 10;
```

```
myClassObj2.myData = 20;
```

```
alert( myClassObj1.myData );
```

```
alert( myClassObj1.myString );
```

```
alert( myClassObj2.myData );
```

```
alert( myClassObj2.myString );
```

```
myClassObj1.myString = "Obj1: Hello World";
```

```
myClassObj2.myString = "Obj2: Hello World";
```

```
// displays: 10
```

```
// displays: "Obj1: Hello World"
```

```
// displays: 20
```

```
// displays: "Obj2: Hello World"
```

Các đối tượng sẽ có các thuộc tính myData và myString

Mô hình đối tượng

- Khai báo phương thức cho lớp

```
function MyClass() {  
    this.myData = 5;  
    this.myString = "Hello World";  
    this.ShowData = DisplayData;    //Khai báo phương thức ShowData()  
    this.ShowString = DisplayString; //Khai báo phương thức ShowString()  
}  
  
function DisplayData() { alert( this.myData ); }  
function DisplayString() { alert( this.myString ); }  
  
var myClassObj1 = new MyClass();  
var myClassObj2 = new MyClass();  
  
myClassObj1.myData = 10;    myClassObj1.myString = "Obj1: Hello World";  
myClassObj2.myData = 20;    myClassObj2.myString = "Obj2: Hello World";  
myClassObj1.ShowData();    // displays: 10  
myClassObj1.ShowString();  // displays: "Obj1: Hello World"  
myClassObj2.ShowData();    // displays: 20
```


Mô hình đối tượng

- Tính đóng gói (encapsulation)

```
function MyClass(){  
    var m_data = 5;  
    this.SetData = SetData;  
    this.ShowData = DisplayData;  
  
    function DisplayData() { alert( m_data ); }  
    function DisplayText() { alert( m_text ); return; }  
    function SetData( myVal ) { m_data = myVal; }  
    function SetText( myText ) { m_text = myText; }  
}
```

Tất cả thuộc tính và phương thức được gói gọn trong khai báo MyClass

```
var myClassObj1 = new MyClass();  
var myClassObj2 = new MyClass();  
myClassObj1.SetData( 10 );  
myClassObj2.SetData( 20 );  
myClassObj1.ShowData();  
myClassObj1.ShowText();  
myClassObj2.ShowData();  
myClassObj2.ShowText();  
  
myClassObj1.SetText( "Obj1: Hello World" );  
myClassObj2.SetText( "Obj2: Hello World" );  
// displays: 10  
// displays: "Obj1: Hello World"  
// displays: 20  
// displays: "Obj2: Hello World"
```

Regular expression

- Biểu thức quy tắc là một chuỗi mô tả một bộ các chuỗi khác, theo quy tắc cú pháp nhất định.
- Biểu thức quy tắc thường được dùng trong các trình biên tập văn bản, các tiện ích tìm kiếm và xử lý văn bản dựa trên mẫu quy định.
- Nhiều ngôn ngữ lập trình cũng được hỗ trợ biểu thức quy tắc trong việc xử lý chuỗi (Perl, PHP, Java, C#, JavaScript).

Regular expression

- Tạo đối tượng Regular Expression
 - **Cách 1:** `/pattern/flags`
 - Ví dụ: `var objRegex = /ab+c/I`
 - Cách này sử dụng khi regular expression giữ nguyên không thay đổi từ lúc tạo cho đến lúc sử dụng.
 - **Cách 2:** sử dụng hàm tạo của RegExp Object
 - Cú pháp: `new RegExp("pattern", "flags")`
 - Ví dụ: `var objRegex = new RegExp("ab+c", "i")`
 - Cách này sử dụng khi regular expression có thể bị thay đổi hay không biết chính xác khi tạo hay lấy từ nguồn dữ liệu khác.

Regular expression

- Giá trị Flags

Ký tự	Mô tả	Ký tự	Mô tả
g	global match	m	xem chuỗi là text có nhiều dòng
i	ignore case	s	xem chuỗi là text chỉ có một dòng
gi	Cả g và i	x	bỏ qua khoảng trắng trong pattern

Chuỗi "javascript" và "JavaScript" được thoả (match) với regexp là `/JavaScript/i`

Chuỗi "abc" được thoả với regexp là `/a b c/x`

Chuỗi "ABCDEFABCGHIABCD" với regexp là `/ABC/g` sẽ cho ra kết quả là "ABC", "ABC", "ABC" (nếu không có flag là g, chỉ cho ra kết quả là "ABC")

Regular expression

- Các phương thức sử dụng trong regexp.

P.thức	Kiểu	Mô tả
exec	RegExp	Tìm kiếm chuỗi thoả điều kiện (được quy định trong pattern). Trả về mảng các thông tin thoả điều kiện
test	RegExp	Kiểm tra chuỗi có thoả điều kiện. Trả về true/false
match	String	Tìm kiếm chuỗi thoả điều kiện. Trả về mảng các thông tin thoả hay trả về null nếu không thoả
search	String	Tìm vị trí thoả điều kiện trong chuỗi. Trả về vị trí thoả điều kiện hay trả về -1 nếu không có vị trí nào thoả
replace	String	Tìm và thay thế chuỗi thoả điều kiện bằng chuỗi khác
split	String	Sử dụng regexp hay chuỗi cố định để tách chuỗi ra thành một mảng chứa các chuỗi con

Regular expression

- Cú pháp của pattern.

Ký tự	Ý nghĩa
\	Ký tự escape (ký tự phía sau ký tự / nếu trùng với các ký tự được sử dụng để thiết lập Pattern sẽ được xem là ký tự bình thường)
^	So khớp (match) ký tự tại vị trí đầu dòng. Ví dụ /^A/ sẽ không khớp với 'A' trong "an A", nhưng khớp 'A' trong "An A"
\$	So khớp ký tự tại vị trí cuối dòng. Ví dụ /t\$/ không khớp với 't' trong "eater" nhưng khớp với 't' trong "eat"
*	So khớp sự xuất hiện 0 hay nhiều lần của ký tự trước dấu *. Ví dụ /bo*/ khớp với 'boooo' trong "A ghost boooooed" và 'b' trong "A bird warbled" nhưng không khớp với trong "A goat grunted"

Regular expression

Ký tự	Ý nghĩa
+	So khớp sự xuất hiện 1 hay nhiều lần của ký tự trước dấu +. Tương đương với $\{1,\}$. Ví dụ <code>/a+/</code> khớp với 'a' trong "candy" và tất cả ký tự 'a' trong "caaaaaaandy"
?	So khớp sự xuất hiện 0 hay 1 lần của ký tự trước dấu ?.
.	So khớp mọi ký tự đơn bất kỳ không phải là ký tự newline '\n'. Ví dụ <code>/.n/</code> khớp với 'an' và 'on' trong "nay, an apple is on the tree", nhưng không khớp với 'nay'
(x)	So khớp 'x' và lưu lại kết quả so khớp (là các chuỗi con – substring) vào mảng kết quả
x y	So khớp x hay y
{n}	So khớp ký tự trước {n} xuất hiện chính xác n lần (với n là số nguyên)

Regular expression

Ký tự	Ý nghĩa
{n,}	So khớp ký tự trước {n,} xuất hiện ít nhất n lần (với n là số nguyên)
{n,m}	So khớp ký tự trước {n,m} xuất hiện ít nhất n lần và nhiều nhất là m lần (với n, m là số nguyên)
[xyz]	So khớp với các ký tự thuộc tập xyz. Có thể dùng dấu – để định nghĩa 1 đoạn các ký tự liên tiếp của một tập. Ví dụ [0-9], [a-z], [A-Z]
[^xyz]	So khớp các ký tự không thuộc tập xyz. Có thể dùng dấu – để định nghĩa 1 đoạn các ký tự liên tiếp của một tập.
[\\b]	So khớp ký tự backspace
\\b	So khớp với ký tự ranh giới (word boundary) là ký tự đầu dòng hay ký tự khoảng trắng. Ví dụ /\\bn\\w/ khớp với 'no' in "noonday"; /\\wy\\b/ khớp với 'ly' trong "possibly yesterday."

Regular expression

Ký tự	Ý nghĩa
<code>\B</code>	So khớp với ký tự không là ký tự ranh giới (non-word boundary). Ví dụ <code>/w\Bn/</code> khớp với 'on' trong "noonday", và <code>/y\Bw/</code> khớp với 'ye' trong "possibly yesterday."
<code>\cX</code>	So khớp với ký tự điều kiện X có trong chuỗi. Ví dụ <code>/cM/</code> khớp với ký tự control-M trong chuỗi
<code>\d</code>	So khớp một ký tự số (tương tự <code>[0-9]</code>)
<code>\D</code>	So khớp một ký tự không là ký tự số (tương tự <code>[^0-9]</code>)
<code>\f</code>	So khớp ký tự form-feed (ký tự có mã ASCII là 12 = <code>\x0C</code> hex)
<code>\n</code>	So khớp với ký tự line feed (ký tự có mã ASCII là 10 = <code>\x0A</code> hex)
<code>\r</code>	So khớp với ký tự về đầu dòng (mã ASCII là 13 = <code>\x0D</code> hex)
<code>\s</code>	So khớp với space character, gồm có khoảng trắng, tab, form feed, line feed (tương tự với <code>[f\n\r\t\v]</code>)

Regular expression

Ký tự	Ý nghĩa
\S	So khớp với các ký tự không là space character ([^\f\n\r\t\v])
\t	So khớp với ký tự tab (ký tự có mã ASCII là 9 = \x09 hex)
\v	So khớp với ký tự vertical tab (mã ASCII là 11 = \x0B hex)
\w	So khớp với mọi ký tự alphanumeric và ký tự gạch dưới (tương tự [A-Za-z0-9_])
\W	So khớp với các ký tự không là alphanumeric (tương đương [^A-Za-z0-9_]). Ví dụ /\W/ khớp với '%' trong "50%."
\n	Tham khảo lại chuỗi con thứ n trong cặp dấu ngoặc đơn (với n là số nguyên dương). Ví dụ /apple(,)\sorange\1/ khớp với 'apple, orange,' trong "apple, orange, cherry, peach."
\xhh	So khớp với ký tự có mã ASCII là hh hex (2 chữ số hex)
\uhhhh	So khớp với ký tự có mã ASCII là hhhh hex (4 chữ số hex)

Regular expression

- VD: hàm bỏ khoảng trắng thừa trong chuỗi

```
function trim(str) {  
    if(!str || typeof str !== 'string')  
        return null;  
    return str.replace(/^[\s]+/, "").replace([\s]+$/, "").replace([\s]{2,}/, '  
'  
});  
}
```

Xén khoảng trắng cuối chuỗi

Xén khoảng trắng đầu chuỗi

Xén khoảng trắng giữa chuỗi

- VD: hàm kiểm tra chuỗi là một con số không

```
function isNumber(number) {  
    var regexp = /^[0-9]+$/;  
    return regexp.test(trim(number));  
}
```

Regular expression

- VD: tìm tất cả các email có trong đoạn văn bản, kết quả đưa vào mảng

```
var text = "This text has some email addresses foo@bar.com. This text  
has some email addresses foobar@foobar.com.au";
```

```
var regexp = /(\\w[-._\\w]*\\w@\\w[-._\\w]*\\w\\.\\w{2,3})/g;  
var arrEmail = text.match(regexp);
```

```
for (var i=0; i < arrEmail.length; i++)  
    document.write(arrEmail[i] + "<br />");
```

```
foo@bar.com  
foobar@foobar.com.au
```

Kết quả chạy đoạn mã

Có thể truy cập vào trang <http://regexlib.com> để tham khảo một số RegExp định nghĩa sẵn (email, URL, number, string, address, phone, date, time...)