Tutorial tự học Javascript được coppy từ nguồn freetuts

# **Bài 01: Javascript là gì? Viêt ứng dụng javascript đầu tiên**

Trong loạt series này chúng ta sẽ tìm hiểu ngôn ngữ Javascript từ căn bản đến nâng cao nhằm giúp các bạn newbie hoặc những bạn đã có kiến thức Javascript rồi nhưng muốn học để bổ trợ thêm. Vì dạng viết text nên nội dung sẽ hơi khó diễn tả và khó trình bày nên các bạn cố gắng đọc từng chữ và thực hành theo demo nhé.

Trong bài này chúng ta sẽ tìm hiểu khái niệm Javascript là gì? ứng dụng của Javscript trong thực tế và viết một ứng dụng javascript đầu tay. Điều kiện tiên quyết để học ngôn ngữ này là  bạn phải biết một số ngôn ngữ khác  như PHP, C++, C nhé.

## **1. Javascript là gì?**

Javascript là một ngôn ngữ lập trình kịch bản dựa vào đối tượng phát triển có sẵn hoặc tự định nghĩa ra, javascript được sử dụng rộng rãi trong các ứng dụng Website. Javascript được hỗ trợ hầu như trên tất cả các trình duyệt như Firefox, Chrome, ... thậm chí các trình duyệt trên thiết bị di động cũng có hỗ trợ.

Nếu bạn đã biết qua về HTML5 thì bạn thấy có các khái niệm như sessionStore hay localStore, đây là hai đối tượng được tạo nên từ Javascript nên rõ ràng trong HTML5 cũng có sử dụng nó. Với những ứng dụng đó thì bạn thấy javascript không thể chết trong các ứng dụng website được.

Hay thậm chí có những ứng dụng Webgame người ta sử dụng javascript để xử lý các thao tác trên Client, nếu không có nó thì thông thường chọn Flash để xây dựng nhưng lại gặp vấn đề load chậm nên có một số người chọn Javascript để làm.

## **2. Các thư viện Javascript đình đám hiện nay**

Những ứng dụng to lớn của Javascript khiến người ta không thể quên nó được. Hiện nay có rất nhiều libraries, Framework được viêt như:

**AngularJS**: Một thư viện dùng để xây dựng ứng dụng Single Page

**NodeJS**: Một thư viện được phát triển phía Server dùng để xây dựng ứng dụng realtime

**Sencha Touch**: Một Framework  dùng để xây dựng ứng dụng Mobile

**ExtJS**: Một Framework dùng xây dựng ứng dụng quản lý (Web Applications)

**jQuery**: Một thư viện rất mạnh về hiểu ứng

...

Còn rất rất nhiều nhưng mình chỉ liệt kê bấy nhieu thôi nhé

## **3. Viết chương trình javascript đầu tiên**

Qua khái niệm javascript là gì? thì bạn thấy tất cả các trình duyệt đều hỗ trợ nên ta không cần phải download thư viện gì cả mà thông qua những đoạn mã script giúp trình duyệt nhận diện được đó là javascript.

**Cặp thẻ <script language="javascript"></scipt>**

Tất cả những đoạn mã Javascript đều phải đặt trong cặp thẻ script như trên, ví dụ:

|  |
| --- |
| <script language="javascript">       alert("Hello World!");  </script> |

**Đặt thẻ script javascript ở đâu?**

Có ba cách đặt sau đây:

**Internal - viết trong file html hiện tại**

Thông thường chúng ta sẽ viết những đoạn mã javascript trên phần head, tuy nhiên đó không phải là điều kiện bắt buộc, nghĩa là bạn có thể đặt ở đâu tùy thích miễn là những đoạn mã đó phải được bao lại bằng thẻ script.

Ví dụ đặt trong thẻ head:

|  |
| --- |
| <html>      <head>          <title></title>          <script language="javascript">              alert("Hello World!");          </script>      </head>      <body>          <!-- Body contents -->      </body>  </html> |

Ví dụ đặt trong thẻ body:

|  |
| --- |
| <html>      <head>          <title></title>      </head>      <body>          <script language="javascript">              alert("Hello World!");          </script>      </body>  </html> |

**External - viết ra một file js khác rồi import vào**

Hoặc bạn viết những đoạn mã javascript ở một file có phần mở rộng là .js, sau đó dùng thẻ script để import vào (giống CSS vậy). Ví dụ file JS của mình có tên là demo.js thì lúc này mình import vào như sau:

|  |
| --- |
| <script language="javascript" src="demo.js"></script> |

Lúc này bên trong file demo.js bạn không đặt thẻ scirpt nữa nhé vì nó là file .js rồi nên trình duyệt tự nhận diện mã JS

**Inline  - viết trực tiếp trong thẻ HTML**

Inline nghĩa là bạn sẽ viết những đoạn mã Javascript trực tiếp trong thẻ HTML luôn. ví dụ dưới đây viết inline  vì đoạn alert(1) được đặt trong sự kiện onclick.

|  |
| --- |
| <input type="button" onclick="alert(1)" value="Click Me"/> |

**Viết chương trình Hello World!**

**Bước 1**: Bây giờ chúng ta thực hành nhé, bạn hãy tạo một file index.html và lưu tại bất kì vị trí nào, desktop hay ổ D, ổ C gì đó thì tùy bạn, miễn là phần mở rộng là .html và gõ nội dung sau vào:

|  |
| --- |
| <html>      <head>          <title></title>      </head>      <body>          <input type="button" value="Click Me"/>      </body>  </html> |

Chạy lên bằng Firefox hoặc Chrome bạn thấy xuất hiện một button. Bây giờ ta sẽ viết ứng dụng khi click vào button đó thì sẽ xuất hiện một thông báo **Hello World** nhé. Nhưng có lưu ý với bạn là button đó mình có đặt id=clickme

**Bước 2**: Viết javascript khi click vào button có id="clickme" thì thông báo lên màn hình.

|  |
| --- |
| <html>      <head>          <title></title>      </head>      <body>          <input type="button" id="clickme" value="Click Me"/>          <script language="javascript">            // Lấy element có id=clickme lưu vào biến button          var button = document.getElementById('clickme');            // Khi click vào element chứa trong button thì thực hiện một function,          // bên trong function thông báo lên Hello World!          button.addEventListener('click', function(){              alert('Hello World!');          });          </script>      </body>  </html> |

Lưu ý là bạn phải đặt đoạn mã javascript bên dưới button như trong demo nhé, nếu bạn đặt ở trên là sẽ bị lỗi đấy, lý do tại sao thì trong các bài tiếp theo mình sẽ giải thích.

Vậy là xong rồi đấy, bây giờ thì bạn biết **javascript là gì?** rồi phải không nào.

## **4. Lời kết**

Bài này mình muốn giới thiệu đến các bạn khái niệm javascript là gì và đồng thời liệt kê ra những thư viện, Framework được xây dựng từ javascript để các bạn thấy quy mô của nó như thé nào. Chúng ta cũng đã làm một ví dụ nho nhỏ sử dụng hàm getElementById và addEventListener, nếu bạn không hiểu hai hàm này thì cứ từ từ nhé, các bài sau ta sẽ nói về nó nhiều hơn.

**Bài 02: Biến và khai báo biến trong javascript**

Trong bài này ta sẽ tìm hiểu các khai báo biến và gán giá trị cho biến trong javascript. , đồng thời học thêm cách dùng hàm document.write để in giá trị biến ra ngoài trình duyệt . Việc khai báo biến trong javascript cũng giống như khai báo biến trong php vậy, tuy cú pháp nó không giống nhau nhưng về cách gán giá trị thì y chang.

**1. Khai báo biến trong javascript**

Để khai báo một biến ta sử dụng từ khóa  var tenbien. Ví dụ:

|  |
| --- |
| var username; |

Tuy nhiên không phải muốn đặt tên biến gì cũng được mà phải tuân theo quy tắc sau đây:

* Tên biến phải là các chữ không dấu viết  hoa hoặc viết thường, các chữ số từ 0-9 và dấu gạch dưới (\_).
* Tên biến bắt đầu phải là chữ hoặc dấu gạch dưới (\_), nếu bắt đầu bằng số là sai
* Tên biến có thể đặt dài hay ngắn tùy vào lập trình viên.

**Ví dụ**:

|  |
| --- |
| // Đúng  var username;  // Đúng  var \_username;  // Đúng  var \_\_username;  // Đúng  var username90;  // SAI  var 90thehalfheart; |

Ngoài cách khai báo trên ta còn có cách khai báo biến liên tiếp cách nhau bởi dấu phẩy, ví dụ:

|  |
| --- |
| var username, \_username, \_\_username, username90; |

**2. Gán giá trị cho biến trong javascript**

Để gán giá trị cho biến ta dùng dấu bằng (=) để gán vế phải vào vế trái, ví dụ gán giá trị**thehalfheart** vào biến **username** thì ta viết là username = 'thehalfheart'.

Có hai cách gán thông dụng đó là vừa khai báo biến vừa gán giá trị và khai báo rồi mới gán giá trị:

**Ví dụ vừa khai báo vừa gán giá trị**:

|  |
| --- |
| var username = 'thehalfheart'; |

**Ví dụ khai báo xong mới gán giá trị**:

|  |
| --- |
| var username;  username = 'thehalfheart'; |

**3. Gán kiểu giá trị cho biến**

Giống như php, để xác định biến có kiểu dữ liệu là gì thì ta dựa vào giá trị mà nó đang có, có nghĩa khi bạn gán một con số không có dấu chấm động thì nó sẽ là kiểu INT, nếu gán một chuỗi thì là kiểu String.

**Ví dụ**:

|  |
| --- |
| // Biến website đag kiểu String  var website = 'thehalfehart';  // Biến website chuyển sang kiểu INT  website = 12;  // Biến website chuyển sang kieeur float  website = 12.5; |

Trong Javascript bạn sẽ có các kiểu dữ liệu thông dụng như kiểu chuỗi (*String*), số (*Number*), mảng (*Array*), đối tượng (*Object*). Chi tiết từng kiểu dữ liệu chúng ta sẽ được học trong các bài tiếp theo.

**4. In giá trị của biến ra trình duyệt**

Để in giá trị của biến hoặc một chuỗi nào đó ra trình duyệt thì ta sử dụng hàm document.write(value).

**Ví dụ**:

|  |
| --- |
| <html>      <head>          <title></title>      </head>      <body>          <script language="javascript">              var website = 'freetuts.net';              document.write(website);          </script>      </body>  </html> |

**Lời kết**

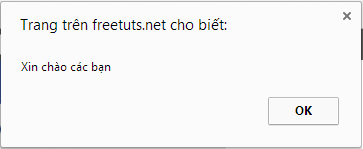
Bài này mình dừng ở đây, hy vọng qua những ví dụ trên giúp bạn hiểu được cách sử dụng biến căn bản trong javascript, tuy nói là căn bản nhưng nó là nền tảng đấy nhé các bạn, hãy xem các ví dụ và thực hành theo, kết hợp nhuần nhuyễn giữa các phần để tự đưa ra những ví dụ cho riêng mình.

**Bài 03: Hàm alert() - confirm() - prompt() trong javascript**

Hôm nay chúng ta sẽ tìm hiểu đến ba hàm rất thông dụng và hữu dụng trong javascript đó là hàm alert(), confirm() prompt(), đây là ba hàm thường được sử dụng để thông báo và lấy thông tin của người dùng để xử lý. Tuy nhiên trong thực tế thời này người ta đã tự viết ra những plugin bằng javascript và HTML để thay thế cho hay hàm này vì mặc định nó thông báo nhìn hơi chuối :D.

**1. Hàm alert() trong javascript**

Hàm alert() có nhiệm vụ in một thông báo popup, nó có một tham số truyền vào và tham số này chính là nội dung ta muốn thông báo với người dùng.



**Ví dụ**: Viết chương trình khi click vào button thì xuất hiện một thông báo.

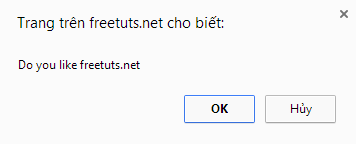
Bài này ta đã làm ở ví dụ bài tìm hiểu [javascript là gì](http://freetuts.net/javascript-la-gi-viet-ung-dung-javascript-dau-tien-263.html) rồi nên mình sẽ viết lại nó hơi khác chút xíu, đó là mình sẽ dùng sự kiện onclick thay vì dùng hàm addEventListener như ở bài trước nhé.

|  |
| --- |
| <html>    <head>    </head>    <body>      <input type="button" onclick="alert('Xin chào các bạn')" value="Click Me"/>    </body>  </html> |

Các bạn thấy mình đã code sử dụng hàm alert() trong sự kiện onclick đấy, đây là cách code inline mà chúng ta đã học ở bài trước rồi nhé.

**2. Hàm confirm() trong javascript**

Hàm confirm() cũng sẽ xuất hiện một thông báo popup nhưng nó có thêm hai sự lựa chọn là Yes và No, nếu người dùng chọn Yes thì nó trả về TRUE và ngược lại nếu chọn NO thì nó sẽ trả về FALSE. Nó cũng có một tham số truyền vào và tham số này chính là nội dung thông báo.



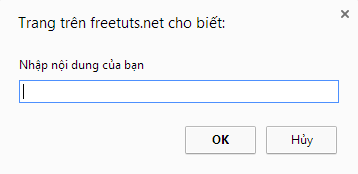
**Ví dụ**: Viết chương trình hiển thị thông báo  xác nhận và dùng hàm alert() để hiển thị kết quả người dùng chọn.

Vì chúng ta chưa học hàm nên mình sẽ không viết dài dòng mà sẽ viết chạy trực tiếp khi load trang, nghĩa là khi trang web được chạy lên thì thông báo nó hiển thị luôn chứ không cần phải click vào button nhé.

|  |
| --- |
| <html>      <head>          <title></title>          <script language="javascript">              confirm("Do you like freetuts.net");          </script>      </head>      <body>        </body>  </html> |

**3. Hàm prompt() trong javascript**

Hàm prompt() dùng  để lấy thông tin từ người dùng, gồm có hai tham số truyền vào là nội dung thông báo và giá trị ban đầu. Nếu người dùng không nhập vào thì giá trị nó sẽ trả về là NULL



**Ví dụ**: Viết chương trìn lấy thông tin tên của người dùng

Ta sẽ [khai báo biến](http://freetuts.net/bien-va-khai-bao-bien-trong-javascript-265.html) lưu trữ kết quả và kết hợp sử dụng hàm alert() để thông báo kết quả người dùng nhập vào.

|  |
| --- |
| <html>      <head>          <title></title>          <script language="javascript">              var t = prompt("Nhập tên của bạn", '');              alert(t);          </script>      </head>      <body>        </body>  </html> |

**Lời kết**

Ba hàm alert() - confirm() - prompt() trong javascript rất hay sử dụng nên các bạn hãy gắng nhớ nó nhé, vì nội dung chúng ta chưa học nhiều nên ví dụ vẫn chưa thấy hay lắm, khi nào ta học đến hàm, events hay selector trong javascript thì bạn sẽ thấy nó tuyệt vời.

**Bài 04: Các toán tử toán học và toán tử gán trong javascript**

Bài này mình không trình bày các khái niệm như "toán tử là gì? biểu thức là gì" vì các bạn đã học PHP hoặc là một ngôn ngữ như C, C++ rồi, mình chỉ hướng dẫn một vài thao tác liên quan đến các phép toán thường hay sử dụng trong javascript thôi nhé.

**1. Toán tử toán học trong javascript**

Thông thường chúng ta sử dụng toán tử toán học như cộng, trừ, nhận, chia, ... để thực xử lý thay đổi giá trị các biến trong javascript. Sau đây là bảng danh sách các toán tử hay dùng.

| **Toán tử** | **Mô tả** |
| --- | --- |
| **+** | Phép cộng. Nếu là chuỗi thì nó sẽ thực hiện thao tác nối chuỗi, còn nêu là số thì nó sẽ cộng hai số lại. Ví dụ:   |  | | --- | | var a = 20;  var b = 5;  // Kết quả biến c có giá trị 25  var c = a + b; | |
| **-** | Phép trừ và dùng với number. Ví dụ:   |  | | --- | | var a = 20;  var b = 5;  // Kết quả biến c có giá trị 15  var c = a - b; | |
| **\*** | Phép nhân và dùng với number. Ví dụ:   |  | | --- | | var a = 20;  var b = 5;  // Kết quả biến c có giá trị 100  var c = a \* b; | |
| **/** | Phép chia và dùng với number. Ví dụ:   |  | | --- | | var a = 20;  var b = 5;  // Kết quả biến c có giá trị 4  var c = a / b; | |
| **%** | Phép chia lấy phần dư, nghĩa là khi chia hai số lại với nhau và kết quả nó sẽ lấy phần dư của phép toán. Ví dụ:  Trường hợp chia dư 0:   |  | | --- | | var a = 20;  var b = 5;  // Kết quả biến c có giá trị 0  // Lý do là a / b dư 0  var c = a % b; |   Trường hợp chia dư khác 0:   |  | | --- | | var a = 22;  var b = 5;  // Kết quả biến c có giá trị 2  // Lý do là a / b = 4 dư 2  var c = a % b; | |
| **++** | Phép tăng giá trị hiện tại lên 1 đơn vị. Phép này có hai cách sử dụng đó là đặt nó trước biến và đặt nó sau biến.   * Trường hợp đứng trước biến thì nó sẽ tăng trước khi lấy giá trị, ví dụ:  |  | | --- | | var c = 12;  alert(++c); // kết quả là 13  alert(c); // kết quả là 13 |  * Trường hợp đứng sau biến thì nó sẽ lấy giá trị rồi tăng lên, ví dụ:  |  | | --- | | var c = 12;  alert(c++); // kết quả là 12  alert(c); // kết quả là 13 | |
| **--** | Phep giảm giá trị hiện tại xuống 1 đơn vị. Phép này cũng có hai cách dùng đó là đặt trước biên và đặt sau biến.   * Trường hợp đứng trước biến thì nó sẽ giảm trước khi lấy giá trị, ví dụ:  |  | | --- | | var c = 12;  alert(--c); // kết quả là 11  alert(c); // kết quả là 11 |  * Trường hợp đứng sau biến thì nó sẽ lấy giá trị rồi giảm xuống, ví dụ:  |  | | --- | | var c = 12;  alert(c--); // kết quả là 12  alert(c); // kết quả là 11 | |

**2. Toán tử gán trong javascript**

Ở các bài trước chúng ta đã sử dụng toán tử gán rất nhiều đó là toán tử gán bằng (=), toán tử này thường hay sử dụng nhiều nhất trong các ngôn ngữ lập trình. Sau đây là danh sách các toán tử gán  mà ta hay dùng trong javascript.

| **Toán tử** | **Ví dụ** | **Mô tả** |
| --- | --- | --- |
| **=** | **x = y** | Gán gí trị của biến x bằng giá trị của biến y, ví dụ:   |  | | --- | | var x = 12;  var y = x; // y = 12 | |
| **+=** | **x += y** | Tương đương với **x = x + y**. Ví dụ:   |  | | --- | | var x = 12;  var y = 10;  // lúc này x = 22  x += y; // tương đương x = x + y | |
| **-=** | **x -= y** | Tương đương với **x = x - y**. Ví dụ:   |  | | --- | | var x = 12;  var y = 10;  // lúc này x = 2  x -= y; // tương đương x = x - y | |
| \***=** | **x \*= y** | Tương đương với **x = x \* y**. Ví dụ:   |  | | --- | | var x = 12;  var y = 10;  // lúc này x = 120  x \*= y; // tương đương x = x \* y | |
| **/=** | **x /= y** | Tương đương với **x = x / y**. Ví dụ:   |  | | --- | | var x = 12;  var y = 10;  // lúc này x = 1.2  x /= y; // tương đương x = x / y | |
| **%=** | **x %= y** | Tương đương với **x = x % y**. Ví dụ:   |  | | --- | | var x = 12;  var y = 10;  // lúc này x = 2;  x %= y; // tương đương x = x % y | |

**3. Lời kết**

Nếu bài này là một ngôn ngữ lập trình căn bản thì mình sẽ trình bày chi tiết hơn nhưng vì các bạn đã học qua ngôn ngữ lập trình khác rồi nên mình chỉ viết ở dạng liệt kê để các bạn dùng để tham khảo và tra cứu thôi nhé. Chúc các bạn học tốt với serie Javascript này.

**Bài 05: Lệnh kiểm tra điều kiện if else trong javascript**

Câu lệnh if else dùng để kiểm tra một mệnh đề nào đó có đúng hay không, nếu đúng thì thực thi những câu lệnh bên trong và ngược lại nếu sai thì nó sẽ bỏ qua những câu lệnh đó. Bây giờ ta sẽ tìm hiểu chi tiết nó nhé.

**1. Lệnh if trong javascript**

**Cú pháp**:

|  |
| --- |
| if (condition){      // statment  } |

Trong đó condition là mệnh đề điều kiện và luôn luôn phải có một trong hai giá trị là true/falsehoặc tương đương (1 => true, 0 => false).

**Ví dụ**:

|  |
| --- |
| var a = 12;  var b = 12;  if (a == b){      alert('a và b bằng nhau');  } |

Kết quả sẽ có thông báo popup "*a và b bằng nhau*".

Trường hợp nếu các câu lệnh bên trong là câu lệnh đơn ( chỉ có một lệnh) thì ta có thể bỏ cặp dấu ngoặc nhọn, với ví dụ trên thì ta sẽ viết lại như sau:

|  |
| --- |
| var a = 12;  var b = 12;    if (a == b)      alert('a và b bằng nhau'); |

**2. Lệnh else trong javascript**

Lệnh else sẽ được thực thi nếu lệnh if không thỏa, như vậy khi dùng lệnh else thì bắt buộc phải có một lệnh if trước nó.

**Ví dụ:**

|  |
| --- |
| var a = 12;  var b = 10;  if (a == b){      alert('a và b bằng nhau');  } else {      alert('a và b khác nhau');  } |

Kết quả sẽ chạy lênh else tại vì lệnh if không thỏa.

**3. Kết hợp nhiều lệnh if else trong javascript**

Ta có thể kết hợp nhiều câu lệnh if để xử lý, cú pháp của nó cũng không có gì khác.

**Ví dụ:**

|  |
| --- |
| var a = 12;  if (a > 12) {      alert('a > 12');  } else if (a < 12) {      alert('a < 12');  } else {      alert('a = 12');  } |

Kết quả sẽ chạy lệnh else cuối cùng tại vì các lệnh if bên trên không thỏa điều kiện. Bây giờ bạn thử đổi giá trị của a thành a = 10 thì câu lệnh else if sẽ được thực hiện.

**4. Lệnh if else lồng nhau trong javascript**

Như các ngôn ngữ khác ta có thể lồng các câu lệnh if lại với nhau, nghĩa là bên trong câu lệnh if sẽ chứa câu lệnh if khác.

**Ví dụ:**

|  |
| --- |
| var a = 13;  // Nếu a > 12  if (a > 12) {      // Khai báo biến b      var b = 20;      // Nếu a bằng b      if (a == b) {          alert(' a = b ');      } else { // ngược lại a khác b          alert(' a != b ');      }  } |

**5. Lời kết**

Quá đơn giản phải không nào, mình không giải thích nhiều lắm vì các bạn đã biết lập trình rồi và cũng thao tác với câu lệnh if else khá nhiều rồi. Còn nếu bạn chưa biết gì về lập trình thì mình nghĩ bạn nên tìm hiểu một ngôn ngữ khác trước khi quay sang đây nhé.

**Bài 06: Vòng lặp for trong javascript**

Vòng lặp for trong javascript thường dùng để lặp một mảng hoặc một khoảng (min max) nào đó nhằm mục đích xử lý giải quyết vấn đề cho bài toán, vì ta đang học căn bản nên trong bài này mình chỉ làm những bài tập đơn giản liên quan đến mảng trong javascript thôi nhé. Trước tiền ta tìm hiểu cấu trúc của nó rồi đi vào vấn đề chính sau:

|  |
| --- |
| var i = 0;  for (i = 0; i < 100; i++){      // Dòng lệnh xử lý vòng lặp  } |

**Trong đó**:

* var i = 0; là [khai báo biến](http://freetuts.net/bien-va-khai-bao-bien-trong-javascript-265.html) điều khiển vòng lặp i
* (i = 0) là điểm bắt đầu lặp (lặp từ 0)
* (i < 100) là điều kiện dừng vòng lặp, nghĩa là lặp nếu i < 100. Bạn có thể dùng một điều kiện bất kì thông qua các [toán tử](http://freetuts.net/cac-toan-tu-toan-hoc-va-toan-tu-gan-trong-javascript-266.html)  miễn là nó trả về **true hoặc false** như (i <= 100, i == 100)
* (i++) là tăng bước nhảy, bạn có thể dùng công thức khác như i+=2, i-=2, i--, ...

Như ví dụ trên thì ta sẽ có 100 vòng lặp từ**0 -> 99**

**1. Một số cách sử dụng vòng lặp for trong javascript**

**Lặp với bước nhảy tăng 1 đơn vị**:

|  |
| --- |
| var i;  // Lặp 10 lần từ 0 -> 9  // Bước nhảy là i++ nên sau mỗi lần lặp i tăng lên 1 đơn vị  for (i = 0; i < 10; i++) {      document.write(i + '<br/>');  } |

**Lặp với bước nhay giảm 1 đơn vị**:

|  |
| --- |
| var i;  // Lặp 10 lần từ 10 -> 1  // Bước nhảy là i-- nên sau mỗi lần lặp i sẽ giảm 1 đơn vị  for (i = 10; i > 0; i--) {      document.write(i + '<br/>');  } |

**Lặp với bước nhay tăng N đơn vị**:

|  |
| --- |
| var i;  var n = 2;  // Lặp 5 lần từ 0 -> 8, bước nhảy là 2  for (i = 0; i < 10; i+=n) {      document.write(i + '<br/>');  } |

**Lặp với bước nhảy giảm N đơn vị**:

|  |
| --- |
| var i;  var n = 2;  // Lặp 5 lần từ 10 -> 2, bước nhảy là -2  for (i = 10; i > 0; i-=n) {      document.write(i + '<br/>');  } |

**Khai báo biến lặp (i) ngay trong vòng lặp**:

Ngoài cách khai báo biến (i) ở ngoài vòng lặp thì ta có thể khai báo trong vòng lặp như ví dụ sau đây:

|  |
| --- |
| for (var i = 0; i < 10; i++) {      document.write(i + '<br/>');  } |

**2. Lặp vô hạn với vòng lặp for trong javascript**

Khi sử dụng vòng lặp for khí khi lặp vô hạn hơn vòng lặp while, do while. Tuy nhiên nếu bạn dùng sai thì sẽ dẫn đến lặp vô hạn đấy

**Dưới đây là một ví dụ lặp vô hạn**:

|  |
| --- |
| for (var i = 0; i < 10; i--) {      document.write(i + '<br/>');  } |

Ta thấy bước nhảy là i-- tức là sau mỗi vòng lặp sẽ giảm - xuống 1 đơn vị trong khi điều kiện dừng là i >= 10, rõ ràng điều kiện này sẽ không bao giờ đúng nên vòng lặp dẫn đến lặp vô hạn.

**3. Vòng lặp for lồng nhau trong javascript**

Khái niệm vòng lặp lồng nhau cũng giống như câu lệnh if else lồng nhau vậy thôi, nghĩa là vòng lặp sẽ nằm trong vòng lặp. Cấu trúc của nó như sau:

|  |
| --- |
| for (var i = 0; i < 10; i++) {      for (var j = 0; j < 10; j++){          // Statment      }  } |

Có một điều lưu ý là hai biến điều khiển vòng lặp phải khác nhau nhé, trong ví dụ trên mình dùng i cho vòng lặp ở ngoài và j cho vòng lặp con phía trong.

**Ví dụ**: Viết chương trình in ra một ma trận 10x10

|  |
| --- |
| for (var i = 0; i <= 9; i++) {      for (var j = 0; j <= 9; j++){          // In ra vị trí của ma trận [i][j]          document.write("(["+i+"]["+j+"])");      }      // Xuống hàng      document.write("<br/>");  } |

**4. Lời kết**

Vòng lặp for khá là đơn giản nên các bạn dễ dàng nắm bắt phải không nào, nó cũng giống như các ngôn ngữ lập trình khác thôi nên để hiểu nó không phức tạp. Hy vọng qua bài vòng lặp for này sẽ giúp bạn học thêm được nhiều điều hơn. Chúc các bạn học tốt.

**Bài 07: Vòng lặp while - do while trong javascript**

Vòng lặp while và do while dùng để lặp với trường hợp tác không biết chính xác số lần lặp là bao nhiêu và trường hợp điều kiện dừng vòng lặp phức tạp, điều này hoàn toàn khác với vòng lặp for. Và khi sử dụng vòng lặp while thì rất dễ bị lặp vô hạn nếu ban không xử lý đúng logic vì thế hãy cẩn thận nhé.

**1. Vòng lặp while trong javascript**

**Cấu trúc vòng lặp while**

|  |
| --- |
| while (condition) {      // do something  } |

Trong đó condition là điều kiện dừng vòng lặp, nếu condition đúng thì vòng lặp sẽ được thực thi cho tới khi condition có giá trị sai. Chính vì vậy nếu condition luôn luôn đúng thì vòng lặp sẽ dẫn tới lặp vô hạn.

**Ví dụ: Dùng vòng lặp while lặp từ 1 tới 10.**

|  |
| --- |
| var i = 1;  while (i <= 10){      document.write(i + '<br/>');      i++; // tăng i lên nếu không sẽ bị lặp vô hạn  } |

Trong ví dụ này biến i có giá trị khởi đầu là  i = 1, điều kiện dừng vòng lặp là i <= 10. Như vậy sau mỗi bước lặp nếu ta không tăng i lên thì vòng lặp sẽ dẫn tới lặp vô hạn.

**Ví dụ: Dùng vòng lặp while lặp từ 10 trở về 1.**

|  |
| --- |
| var i = 10;  while (i >= 1){      document.write(i + '<br/>');      i--; // giảm i xuống nếu không sẽ bị lặp vô hạn  } |

Trường hợp này thì lại khác sau mỗi vòng lặp thì ta phải giảm i xuống 1 đơn vị vì điều kiện lặp là i >= 1.

**Ví dụ: Lặp với điều kiện phức tạp.**

Ta sẽ làm một ví dụ đơn giản đó là sử dụng hàm prompt trong javascript để lấy thông tin từ người dùng, nếu người dùng nhập vào số trong khoảng **1 -> 10** thì dừng, ngược lại thì yêu cầu họ nhập lại.

|  |
| --- |
| // Biến lưu giá trị người dùng nhập vào  var value = null;    // Trong khi giá trị bé hơn 1 hoặc giá trị lớn hơn 10  // thì chạy thân vòng lặp  while (value < 1 || value > 10){      value = prompt("Nhập vào số từ 1 -> 10");  }    // Sau khi nhập đúng thì in ra màn hình  alert("Số bạn vừa nhập là " + value); |

Như vậy trường hợp này ta không hề biết là lặp bao nhiêu lần cả phải không nào :D.

**2. Vòng lặp do while trong javascript**

Khác một chút so vói vòng lặp while là vòng lặp do while sẽ thực thi trường rồi mới kiểm tra điều kiện. Ví dụ bạn chơi trò chơi bốc thăm, lần đầu tiên bạn sẽ bốc nếu cây thăm may mắn thì bốc tiếp và ai được nhiều cây thăm may mắn thì thắng, như vậy có nghĩa là bạn sẽ được bốc 1 lần rồi mới kiểm tra điều kiện cho lần bốc kế tiếp. Từ đó suy ra trong vòng lặp do while sẽ luôn luôn thực thi lặp **ít nhất 1 lần**.

**Cấu trúc vòng lặp do while**

|  |
| --- |
| do {      // some thing  } while (condition); |

Trong đó **condition** là điều kiện để dừng vòng lặp.

**Ví dụ chứng minh vòng lặp do while luôn luôn lặp ít nhất là 1 lần**

Hồi nãy mình nói có vẻ bạn không tin nên mình sẽ làm một ví dụ luôn. Như bạn biết nếu điều kiện **condition** là false thì sẽ không lặp thì bây giờ ta cho nó false luôn để xem có lặp không nhé.

|  |
| --- |
| do {      alert(1);  }  while (false); |

Chạy lên bạn sẽ thấy nó có lặp 1 lần. Bây giờ ta thay bằng vòng lặp while xem có lặp không nhé.

|  |
| --- |
| while (false){      alert(1);  } |

Im re phải không nào :D

**Ví dụ: Sử dụng vòng lặp do while để viết lại chương trình yêu cầu nhập các số từ 1 -> 10 ở ví dụ trong phần vòng lặp while.**

|  |
| --- |
| // Biến lưu giá trị người dùng nhập vào  var value = null;  // Thực thi  do {      value = prompt("Nhập vào số từ 1 -> 10");  } while (value < 1 || value > 10);    // Sau khi nhập đúng thì in ra màn hình  alert("Số bạn vừa nhập là " + value); |

**3. Vòng lặp while - do while lồng nhau**

Cũng giống như trong vòng lặp for, ta có thể lồng nhiều vòng lặp while hoặc do while lại với nhau để viết chương trình.

**Ví dụ: Dùng vòng lặp while xây dựng ma trận 10x10.**

|  |
| --- |
| var i = 0;  // Vòng lặp ngoài  while (i <= 9) {      // Mỗi lần lặp gan j = 0;      var j = 0;        // Lặp nếu j < <=      while (j <= 9) {          // In ra màn hình          document.write('['+i+']['+j+'] ');            // Tăng j lên nếu không sẽ bị lặp vô hạn          j++;      }      // Xuống hàng      document.write('<br/>');        //Tăng i lên nếu không sẽ bị lặp vô hạn      i++;  } |

Tương tự cho vòng lặp do while các bạn tự làm lấy nhé.

**4. Lời kết**

Điểm lưu ý nhất của bài này là so sánh sự khác biệt giữa vòng lặp while và do while trong javascript, cách sử dụng nó và thường khi nào thì sử dụng. Đối với vòng lặp do while thì nó luôn luôn lặp ít nhất 1 lần tại vì nó lặp trường rồi mới kiểm tra điều kiện. Chỉ vậy thôi :D chúc bạn học tốt.

**Bài 08: Hàm và tạo hàm (function) trong javascript**

Hàm là một phương pháp lập trình truyền thống và thường được ứng dụng trong các phương pháp lập trình thủ tục, lập trình hướng module,... Với các ngôn ngữ lập trình bậc cao như C++ thì việc dùng hàm để code ứng dụng là người ta không thích tại vò khó quản lý bảo trì, nhưng đối với javascript và PHP thì ta rất hay dùng hàm.

**1. Định nghĩa hàm (function) trong javascript**

Từ bài 1 tới giờ ta đang code từng đoạn code riêng lẻ và khi cần thì code lại, như vậy giả sử ta cần xử lý vấn đề đó trong 1000 trường hợp thì ta phải code lại 1000 lần nên rất tốn thời gian. Chính vì vậy người ta nghĩ ngay đến Hàm, có nghĩa là sẽ gom một số đoạn code vào một khối xử lý và khi cần thì gọi ra dùng.

Giả sử mình viết chương trình kiểm tra số chẵn hay lẻ như sau:

|  |
| --- |
| var number = 2;  if (number % 2 == 0) {     alert('Số chẵn');  } else {     alert('Số lẻ');  } |

Giả sử mình cần kiểm tra cho 100 số, như vậy mình cứ viết if else cho 100 lần quá nản phải không nào :D. Để giải quyết nó ta tìm hiểu hàm đã nhé.

**Cú pháp tạo hàm trong javascript**

|  |
| --- |
| function name\_of\_function(var1, var2, var3, ...) {      // Some code  } |

**Trong đó**:

* **function**: là từ khóa của javascript nên bắt buộc phải như vậy
* **name\_of\_function**: là tên của function, thông thường chúng ta tạo những tên có ý nghĩa như find\_max, find\_min, ...
* **var1, var2 var3, ...** là các tham số truyền vào hàm. Ví dụ viết hàm kiểm tra số chẵn hay lẽ thì ta sẽ có một tham số đó là số cần kiểm tra.

**Ví dụ: Viết hàm kiểm tra một số chẵn hay lẻ.**

|  |
| --- |
| // Tạo hàm  function check\_number(number) {      if (number % 2 == 0) {          alert(number + ' là số chẵn');      } else {          alert(number + 'Số lẻ');      }  }  // Sử dụng hàm kiểm tra cho 5 số  check\_number(1);  check\_number(2);  check\_number(3);  check\_number(4);  check\_number(5); |

Các bạn thấy mình đã tạo một hàm với tham số truyền vào có tên là number. Như vậy khi sử dụng nếu ta truyền số 1 vào thì lúc này biến number trong hàm sẽ có giá trị là 1, tương tự cho 2, 3,4,5.

Lưu ý với các bạn trong javascript không tồn tại khái niệm con trỏ và tham chiếu trong hàm

**2. Hàm có return và hàm không có return**

Hàm có return là hàm có sử dụng từ khóa return để đặt ở cuối hàm với mục đích trả kết quả về để sử dụng tiếp ở những đoạn code bên ngoài.  Ví dụ ta cần viết một hàm tính tổng của hai số a và b thì hàm này phải trả về giá trị là tổng của hai số a, b. Xem ví dụ sau: [XEM DEMO](http://freetuts.net/editor.html?id=208)

|  |
| --- |
| // Khai báo hàm  function tinh\_tong(a, b) {      // trả về kết quả là a + b      return a + b;  }  // Sử dụng  var so1 = 1;  var so2 = 2;  // truyền so1 và so2 vào hàm  var ketqua = tinh\_tong(so1, so2);  alert(ketqua); |

Hàm không có return là hàm không có sử dụng từ khóa return đặt trong hàm. Ví dụ viết chương trình in ra tổng của hai số a và b.

|  |
| --- |
| // Khai báo hàm  function tinh\_tong(a, b) {      document.write('Tổng là ' + (a + b));  }  // Sử dụng  var so1 = 1;  var so2 = 2;  // truyền so1 và so2 vào hàm  tinh\_tong(so1, so2); |

Như vậy tùy vào mục đích mà ta dùng có return hay không có return. Nhưng thông thường ta sử dụng return ở những trường hợp cần lấy kết quả đó để xử lý tiếp, như ở ví dụ trên đó là mình lấy kết quả để **alert**.

**3. Một số ví dụ tạo hàm trong javascript**

**Ví dụ 1: Viết chương trình kiểm tra một năm có phải là năm nhuận hay không**

Năm nhuận là năm chia hết cho 4, nếu chia hết cho 100 thì nó phải chia hết cho 400. Đây là định nghĩa năm nhuận còn chính xác hay không thì mình không biết nhé :D vì có trường hợp nó sai.

|  |
| --- |
| // khai báo hàm  function kiem\_tra\_nam\_nhuan(nam) {      // nếu năm chia hết cho 100      // thì kiểm tra nó có chia hết cho 400 hay không      if (nam % 100 == 0) {          // nêu chia hết cho 400 thì là năm nhuận          if (nam % 400 == 0) {              alert(nam + ' là năm nhuận');          } else { // ngược lại không phải năm nhuận              alert(nam + ' không phải năm nhuận');          }      } else if (nam % 4 == 0){ // trường hợp chia hết cho 4 thì là năm nhuận          alert(nam + ' là năm nhuận');      } else { // cuối cùng trường hợp không phải năm nhuận          alert(nam + 'không phải là năm nhuận');      }  }  // sử dụng  kiem\_tra\_nam\_nhuan(4); |

**Ví dụ 2: thực hiện lại ví dụ trên nhưng sử dụng return để trả kết quả vè, nếu true thì là năm nhuận, false thì không phải năm nhuận.**

|  |
| --- |
| // khai báo hàm  function kiem\_tra\_nam\_nhuan(nam) {      // nếu năm chia hết cho 100      // thì kiểm tra nó có chia hết cho 400 hay không      if (nam % 100 == 0) {          // nêu chia hết cho 400 thì là năm nhuận          if (nam % 400 == 0){              return true;          } else { // ngược lại không phải năm nhuận              return false;          }      } else if (nam % 4 == 0){ // trường hợp chia hết cho 4 thì là năm nhuận          return true;      } else { // cuối cùng trường hợp không phải năm nhuận          return false;      }  }  // sử dụng  var flag = kiem\_tra\_nam\_nhuan(4);  if (flag){      alert('là năm nhuận');  } else {      alert('không phải là năm nhuận');  } |

Trong ví dụ này thay vì alert trực tiếp trong hàm thì ta trả về kết quả true/false, sau đó kiểm tra kết quả này nếu true thì là năm nhuận, nếu false thì không phải là năm nhuận.

**4. Lời kết**

Qua bài này bạn phải hiểu được cách sử dụng hàm và tạo hàm trong javascript, hiểu được trong javascript không tồn tại khái niệm con trỏ và tham chiếu, hiểu được hai cách sử dụng hàm là hàm có return và hàm không có return. Chúc bạn học tốt!

**Bài 09: Một số ví dụ học javascript căn bản (Phần 1)**

Chúng ta đã học được 8 bài rồi nên trong bài này mình sẽ làm một số ví dụ dùng javascript để viết một số chương trình căn bản để các bạn nắm rõ hơn và ôn lại kiến thức cũ. Nội dung của bài chủ yếu sử dụng kiến thức ở các bài trước nên nếu bạn chư xem thì quay lại và đọc qua nhé.

**1. Bài tập tìm số lớn - nhỏ nhất với javascript**

Ta sẽ viết một chương trình cho người dùng nhập các con số vào cho tới khi không nhập nữa, sau đó sẽ xuất thông tin ra số nào lớn nhất và số nào nhỏ nhất.

Để giải quyết bài này chúng ta sẽ có ý tưởng như sau:

* Khai báo hai biến min và max để lưu giá trị nhỏ nhất và lớn nhất
* Sử dụng hàm [prompt()](http://freetuts.net/ham-alert-confirm-prompt-trong-javascript-264.html) để lấy thông tin. Hàm này nếu ta nhấn hủy thì sẽ trả về NULL
* Sử dụng vòng lặp while để yêu cầu người dùng nhập vào cho tới khi họ hủy. Vì khi click hủy sẽ có giá trị null nên điều kiện dừng của vòng lặp while là NULL
* Cuối cùng dùng lệnh document.write để in ra màn hình.

Các bạn xem code sau: [XEM DEMO](http://freetuts.net/editor.html?id=216)

|  |
| --- |
| // Hai biến lưu min và max  var min = null;  var max = null;  // Biến lưu giá trị người dùng nhập vào  var value = '';  // trong khi người dùng chưa nhấn Hủy  // vì người dùng nhấn hủy tức là value = null,  // thì sẽ dừng vòng lặp  while (value != null) {      // Lấy giá trị      value = prompt("Nhập số, nếu muốn dừng thì không nhập gì");        // Nếu người dùng nhấn hủy thì không thực thi những đoạn code bên trong      if (value != null) {          // Nếu nhập lần đầu thì gán cả hai min và max bằng value          if (min == null) {              min = value;              max = value;          } else {              // ngược lại sẽ kiểm tra gán min và max              if (min > value) {                  min = value;              }              if (max < value) {                  max = value;              }          }      }  }  // In ra trình duyệt  document.write("Min là " + min +", Max là " + max); |

**2. Bài tập giải phương trình bậc nhất với javascript**

Phương trình bậc nhất có dạng ax + b = 0. Như vậy ta có các trường hợp nghiệm như sau:

* Nếu a = 0 và b = 0 thì phương trình vô số nghiệm
* Nếu a # 0 và b = 0 thì nghiệm là x = 0
* Nếu a = 0 và b # 0 thì phương trình vô nghiệm
* Nếu a # 0 và b # 0 thì nghiệm là x = -b/a

Vì ta có hai số a, b là người dùng nhập vào, còn x là nghiệm nên ta sẽ viết chương trình cho người dùng nhập vào a, b rồi tính toán trên hai biến này. Các bạn xem code như sau:

|  |
| --- |
| function giai\_pt\_bac\_nhat(a, b) {      if (a == 0 && b == 0){          alert('Phương trình vô số nghiệm');      } else if (a != 0 && b == 0){          alert('Phương trình có nghiệm x = 0');      } else if (a == 0 && b != 0){          alert("Phương trình vô nghiệm");      } else {          alert('Phương trình có nghiệm x = ' + (-b/a));      }  }  giai\_pt\_bac\_nhat(1,0); |

**3. Bài tập kiểm tra số chính phương với javascript**

**Định nghĩa**: Số chính phương là số có căn bậc hai là một số tự nhiên không có dấu chấm động. Ví dụ 4 là số chính phương vì căn bậc hai của 4 bằng 2. Còn 5 không phải là số chính phương vì căn bậc hai của nó là số có dư  phần dấu chấm động.

Như vậy để giải bài này ta chỉ cần kiểm tra căn bậc hai của nó có tròn hay không, nếu số tròn không dư phần số phẩy ở sau thì là số chính phương. Và để kiểm tra một số có dư hay không thì ta chia nó cho 1 là được, ví dụ 1.2 % 1 = 0.2

Các bạn xem bài giải sau: [XEM DEMO](http://freetuts.net/editor.html?id=218)

|  |
| --- |
| function so\_chinh\_phuong(a) {      // Ta dùng hàm Math.sqrt để lấy căn bậc 2      // sau đó chia lấy dư với 1, nếu bằng 0 thì là số chính phương,      // ngược lại thì không phải là số chính phương      if (Math.sqrt(a) % 1 == 0) {          alert(a + " là số chính phương");      } else {          alert(a + " không phải là số chính phương");      }  }  so\_chinh\_phuong(4);  so\_chinh\_phuong(12);  so\_chinh\_phuong(13); |

**Lời kết:**

Đêm khuya muộn rồi nên mình sẽ dừng ở đây, hy vọng qua ba bài tập javascript căn bản này sẽ giúp bạn ôn lại kiến thức javascript căn bản mà 8 bài trước chúng ta đã được học.

**Bài 10: Thao tác với mảng trong javascript**

Việc xử lý mảng trong javascript cũng quan trọng như xử lý mảng trong PHP vậy, bởi vì khi chúng ta dùng javascript để xử lý DOM thì kết quả mà chúng ta lấy được là một mảng các Element HTML và sẽ lặp element đó rồi xử lý. Chính vì điều này nên hôm nay mình viết một bài xử lý mảng trong javascript luôn để trình bày cho các bạn dễ nắm bắt hơn.

Trước hết chúng ta tìm hiểu khái niệm mảng là gì đã nhé. Mảng là một tập hợp các phần tử lại và mỗi phần tử sẽ được đánh dấu một vị trí trong tập hợp đó. Trong javascript nếu mảng có 10 phần tử thì các phần tử sẽ được đánh dấu từ 0 -9.

**1. Khai báo mảng trong javascript**

Chúng ta có hai cách thông thường để khai báo mảng đó là sử dụng từ khóa new Array() và sử dụng cặp dấu ngoặc vuông ([]).

**Khai báo với từ khóa new Arrray()**

**Cú pháp**:

|  |
| --- |
| var name\_array = new Array();  // Hoặc  var name\_array = new Array(1,2,3); |

Với cách hai thì ta khởi tạo giá trị lúc khai báo luôn.

**Khai báo với cặp dấu ngoặc vuông ([])**

**Cú pháp**:

|  |
| --- |
| var name\_array = [];  // Hoặc  var name\_array = [1,2,3]; |

Tương tự cách hai là ta khai báo và khởi tạo giá trị luôn.

**2. Truy xuất các phần tử trong mảng**

Để truy xuất đến phần tử của một mảng chúng ta dùng cú pháp **tenmang[vitri]**. Ví dụ:

|  |
| --- |
| var t = new Array(1,2,3);  alert(t[0]); // kết quả là 1  alert(t[1]); // kết quả là 2  alert(t[2]); // kết quả là 3 |

Như vậy phần tử đầu tiên sẽ có số chỉ mục là 0, phần tử thứ hai là 1, ... phần tử thứ n là n-1.

**3. In mảng ra trình duyệt và console**

**Hàm array.join**()

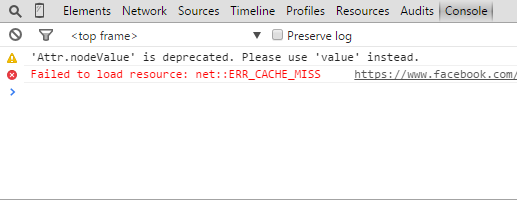
Để hiển thị các phần tử ra ngoài trình duyệt chúng ta sẽ sử dụng hàm array.join(). Ví dụ:

|  |
| --- |
| var t = new Array(1,2,3);  document.write(t.join()); // kết quả 1,2,3  document.write(t.join('-')); // kết quả 1-2-3 |

Như vậy hàm join này được tích hợp vào đối tượng mảng trong javascript và nó có một tham số truyền vào, nếu ta không truyền gì vào thì mặc định nó lấy dấu phẩy để ngăn giữa các giá trị khi in lên trình duyệt.

**Hàm console.log()**:

Ngoài ra chúng ta có thêm một hàm rất hay dùng để debug đó là console.log().  Các bạn hãy cài đặt Firebug trên firefox hoặc dùng chế độ "kiểm tra phần tử" có sẵn trên trình duyệt (chrome tương tự), sau đó chuyển vào mục console như hình dưới đây:

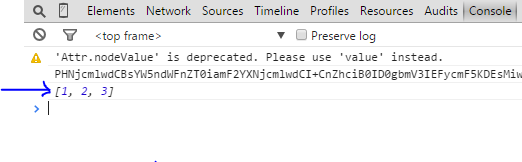


Nêu bạn code bị lỗi cú pháp hoặc một lỗi bất kì liên quan đến javascript thì nó sẽ hiển thị ở ô này nên bạn phải dùng nó để kiểm tra nếu viết mã javascript mà không thấy hoạt động nhé. Như trong hình chữ đỏ là lỗi, chữ vàng là cảnh báo.

Ví dụ:

|  |
| --- |
| var t = new Array(1,2,3);  console.log(t); |

Chạy lên giao diện trong console sẽ như sau:



**4. Sử dụng vòng lặp để lặp mảng trong javascript**

Phần này là quan trọng nhất đấy nhé, chúng ta sẽ học cách sử dụng các vòng lặp đã học để lặp mảng. Nhưng trước tiên chúng ta tìm hiểu cách đếm tổng số phần tử trong mảng như sau:

Để đếm tổng số phần tử của một mảng chúng ta sẽ dùng thuộc tính length của nó. Ví dụ:

|  |
| --- |
| var t = new Array(1,2,3);  alert(t.length);<br> |

**Lặp mảng với vòng lặp for**

Để lặp mảng với [vòng lặp for](http://freetuts.net/vong-lap-for-trong-javascript-270.html) thì chúng ta phải dùng thuộc tính length như trên để đếm tổng số phần tử, sau đó ở mỗi vòng lặp chúng ta sử dụng cú pháp truy xuất đến phần tử của mảng ở phần 2 để xử lý. Ví dụ:

|  |
| --- |
| var name\_array = [1,2,3];  for (var i = 0; i < name\_array.length; i++) {      document.write(name\_array[i]);  } |

**Lặp mảng với vòng lặp while**

Tương tự để lặp với vòng lặp while  chúng ta sẽ khai báo một biến index để lưu vị trí đang lặp. Ví dụ:

|  |
| --- |
| var name\_array = [1,2,3];  var index = 0;  while (index < name\_array.length) {      document.write(name\_array[index]);      index++;  } |

Riêng đối với vòng lặp do while thì không khuyến khích sử dụng để lặp mảng, lý do nó vòng lặp do while luôn luôn lặp ít nhất một lần nên trong trường hợp mảng cần lặp rỗng thì sẽ bị báo lỗi ngay.

**5. Lời kết**

Bài này trọng tâm là cách dùng vòng lặp for để lặp mảng vì khi làm ứng dụng thực tế đa số chúng ta sư dụng vòng lặp for để lặp. Và một lưu ý nữa là không nên dùng vòng lặp do while để lặp mảng nhé, lý do thì trong bài đã đề cập tới rồi. Bài tiếp theo chúng ta tìm hiểu một số hàm xử lý mảng trong javascript

**Bài 11: Các hàm xử lý mảng trong javascript**

Việc xử lý mảng trong javascript đóng vai trò rất quan trọng vì nó được sử dụng khá nhiều trong thực tế.  Chính vì vậy việc biết một số hàm xử lý mảng thông dụng trong javascript sẽ giúp bạn dễ dàng học cũng như tìm hiểu javascript hơn.

Mảng thực chất cũng là một Object (đối tượng) nên thực chất các hàm mà ta hay gọi chính là các phương thức (method) của đối tượng mảng, vì vậy bạn đừng nhầm lẫn giữa hai khái niệm nhé. Trong nội dung bài này mình sẽ gọi là các hàm xử lý mảng để tiện cho việc diễn giải hơn.

**1. Danh sách các hàm xử lý mảng trong javascript**

Phần này chúng ta sẽ tìm hiểu một số hàm có sẵn trong javascript được tích hợp vào array object, chính vì nó chỉ dùng trong array object nên nếu bạn dùng với kiểu dữ liệu khác sẽ bị sai đấy nhé.

**Hàm array.valueOf()**

Hàm này có tác dụng tương tự như hàm array.join() mà ta đã học ở bài trước, có nghĩa là nó sẽ nối các phần tử với nhau vào một chuỗi cách nhau bởi dấu phẩy.

**Ví dụ**:

|  |
| --- |
| var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];  document.write(mang.valueOf()); |

**Hàm array.push()**

Hàm thêm một phần tử vào cuối mảng.

**Ví dụ**:

|  |
| --- |
| var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];  // in mảng  document.write(mang.valueOf());  document.write('<br/>');  // Thêm và in  mang.push("Miễn phí");  document.write(mang.valueOf()); |

**Hàm array.pop()**

Ngược với hàm array.push(), hàm này có tác dụng xóa đi phần tử cuối cùng trong mảng.

**Ví dụ**:

|  |
| --- |
| var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];  // in mảng  document.write(mang.valueOf());  document.write('<br/>');  // Thêm và in  mang.pop();  document.write(mang.valueOf()); |

**Hàm array.shift()**

Hàm xóa phần tử đầu tiên của mảng, sau đó dồn các phần tử phía sau xuống một bậc.

**Ví dụ**:

|  |
| --- |
| var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];  // in mảng  document.write(mang.valueOf());  document.write('<br/>');  // Thêm và in  mang.shift();  document.write(mang.valueOf()); |

**Hàm array.unshift()**

Thêm một phần tử vào vị trí đầu tiên của mảng, đồng thời đẩy các phẩn từ phía sau lên một bậc.

**Ví dụ**:

|  |
| --- |
| var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];  // in mảng  document.write(mang.valueOf());  document.write('<br/>');  // Thêm và in  mang.unshift("Chào Mừng");  document.write(mang.valueOf()); |

**Hàm array.splice()**

Hàm splice() có ba tham số truyền vào như sau: splice(position\_add, num\_element\_remove, value1, value2, ...).

**Trong đó**:

* **position\_add** là vị trí sẽ thêm (vị trí đầu tiên là 0)
* **num\_element\_remove** là số phần tử sẽ xóa (bắt đầu từ position\_add)
* **value1**, **value2**, .. là danh sách các phần tử sẽ được thêm vào sau khi tại vị trí position\_add và sau khi remove num\_element\_remove phần tử.

**Ví dụ 1**: [XEM DEMO](http://freetuts.net/editor.html?id=276)

|  |  |
| --- | --- |
| 1  2  3  4 | var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];    mang.splice(1, 2, 'PHP', 'căn bản ');  document.write(mang.valueOf()); |

Trong ví dụ này thì:

* Vị trí thêm là số 1 (phần tử có giá trị là "lập")
* Xóa 2 phần tử liên tiếp từ vị trí 1 (xóa phần tử "lập" và "trình")
* Thêm hai phần tử "php" và "căn bản" vào

Cuối cùng ta có được một mảng gồm ["Học", "php", "căn bản", "tại", "freetuts.net"].

**Ví dụ 2**:

Trong ví dụ này ta sẽ không thêm phần tử nữa mà sẽ lợi dụng hàm này để xóa đi một số phần tử.

|  |  |
| --- | --- |
| 1  2  3  4 | var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];  // xóa phần tử "lập" và "trình"  mang.splice(1, 2);  document.write(mang.valueOf()); |

**Hàm array.sort()**

Hàm này dùng để sắp xếp các phần tử trong mảng theo thứ tự chữ cái alpha.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];  document.write(mang.valueOf());  document.write('<br/>');    // Sắp xếp lại  mang.sort();  document.write(mang.valueOf()); |

**Hàm array.reverse()**

Hàm đảo ngược các phẩn tử lại. Vị trí đầu sẽ được chuyển xuống cuối mảng và vị trí cuối mảng sẽ được chuyển lên đầu mảng.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=279)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];  document.write(mang.valueOf());  document.write('<br/>');    // Đảo ngược vị trí của các phần tử  mang.reverse();  document.write(mang.valueOf()); |

**Hàm array.concat()**

Hàm dùng để nối hai mảng với nhau và trả về một mảng gồm tổng số phần tử của hai mảng đó.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=280)

|  |  |
| --- | --- |
| 1 | var mang1 = ["Học", "lập", "trình"];<br>var mang2 = ["tại", "freetuts.net"];<br><br>// Nối mảng<br>var mang\_con = mang1.concat(mang2);<br>document.write(mang\_con.valueOf()); |

**Hàm array.slice()**

Hàm dùng để lấy một số phần tử con trong mảng. Có hai tham số truyền vào như sau: slice(start, end).

**Trong đó**:

* **start**: là vị trí bắt đầu
* **end**: là vị trí kết thúc

Lưu ý: Để dễ hiểu thì start sẽ phần tử đầu tiên là 0 và end sẽ tính phần tử đầu tiên là 1. Chính vì vậy bạn sẽ phải cộng thêm 1 ở end thì mới lấy đúng phần tử mong muốn

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=281)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];    // Lấy phần tử "tại" và "freetuts.net"  var mang\_moi = mang.slice(3, 5);    // In ra thử  document.write(mang\_moi.valueOf()); |

Trường hợp bạn muốn lấy từ vị trí nào đó đến cuối mảng thì bạn sẽ truyền một tham số thôi.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=282)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <!DOCTYPE html>  <html>      <body>          <script language="javascript">              var mang = ["Học", "lập", "trình", "tại", "freetuts.net"];                // Lấy phần tử "tại" và "freetuts.net"              var mang\_moi = mang.slice(3);                // In ra thử              document.write(mang\_moi.valueOf());          </script>      </body>  </html> |

**2. Lời kết**

Trên là danh sách **các hàm xử lý mảng trong javascript** mà ta hay sử dụng nhất, vẫn còn khá nhiều hàm nên mình không thể liệt kê hết được. Sau này học nâng cao lên bạn có thể tự định nghĩa thêm các hàm riêng cho mình bằng cách sử dụng prototype.

**Bài 12: DOM là gì? Các loại DOM trong Javascript**

Javascript là một ngôn ngữ được sử dụng trong các trình duyệt Browser nên nó đóng một vai trò khá quan trọng trong các ứng dụng website. Và nhiệm vụ của Javascript là thao tác với các tài liệu HTML kết hợp với các cú pháp riêng của nó để tạo nên sự ảo diệu của trang web.

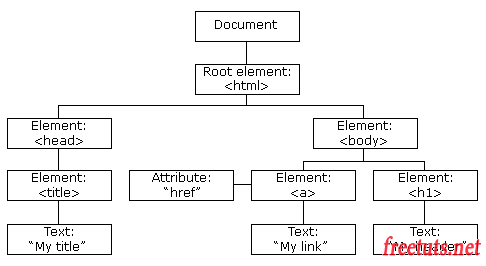
Để thao tác được với các thẻ HTML thì nó phải thông qua một cơ chế ta gọi là DOM và ta hay gọi là **D**ocument **O**bject **M**odel. Vậy thì DOM là gì thì chúng ta cùng tì hiểu nó nhé.

**1. DOM là gì?**

DOM là viết tắt của chữ **D**ocument **O**bject **M**odel, dịch tạm ra là mô hình các đối tượng trong tài liệu HTML.

Như các bạn biết trong mỗi thẻ HTML sẽ có những thuộc tính (Properties) và có phân cấp cha - con với các thẻ HTML khác. Sự phân cấp và các thuộc tính của thẻ HTML này ta gọi là[selector](http://freetuts.net/selector-la-gi-tim-hieu-css-selector-can-ban-345.html) và trong DOM sẽ có nhiệm vụ xử lý các vấn đề như đổi thuộc tính của thẻ, đổi cấu trúc HTML của thẻ, ...

Bạn có thể tham khảo hình vẽ dưới đây để hiểu rõ hơn về DOM.



Trong hình thì tất cả các thẻ HTML sẽ được quản lý trong đối tượng **document**, thẻ cao nhất là thẻ **html**, tiếp theo là phân nhánh **body**và **head**. Bên trong **head**thì có những thẻ như **style**,**title**, ... và bên trong **body**thì là vô số các thẻ HTML khác.

Như vậy trong Javascript để thao tác với các thẻ HTML ta phải thông qua đối tượng document nhé các bạn, để cho nóng thì mình làm một ví dụ trước rồi tìm hiểu sau :D.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=283)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <html>    <body>      <h1 id="main-content"></h1>      <script language="javascript">        document.getElementById("main-content").innerHTML = "Chào mừng các bạn đến với website học lập trình online freetuts.net"      </script>    </body>  </html> |

Trong ví dụ này mình có sử dụng một đoạn code xử lý javascript như sau:

|  |  |
| --- | --- |
| 1 | document.getElementById("main-content").innerHTML = "Chào mừng các bạn đến với website học lập trình online freetuts.net" |

Và đoạn code này có ý nghĩa rằng tìm thẻ có id="main-content"và gán nội dung HTML bên trong của thẻ này là dòng chữ "*Chào mừng các bạn đến với website học lập trình online freetuts.net*".

**2. Các thể loại DOM trong Javascript**

Việc xử lý và làm việc với đối tượng HTML rất phức tạp và đa dạng, chính vì vậy javascript có cung cấp cho chúng ta nhiều phương thức, đối tượng và mỗi thành phần như vậy sẽ có những nhiệm vụ riêng biệt. Sau đây mình sẽ liệt kê danh sách chia nhóm và trong những bài học tiếp theo chúng ta sẽ cùng tìm hiểu nó nhé.

**Danh sách chia nhóm DOM**:

* DOM document: có nhiệm vụ lưu trữ toàn bộ các thành phần trong tài liệu của website
* DOM element: có nhiệm vụ truy xuất tới thẻ HTML nào đó thông qua các thuộc tính như tên class, id, name của thẻ HTML
* DOM HTML: có nhiệm vụ thay đổi giá trị nội dung và giá trị thuộc tính của các thẻ HTML
* DOM CSS: có nhiệm vụ thay đổi các định dạng CSS của thẻ HTML
* DOM Event: có nhiệm vụ gán các sự kiện như onclick(), onload() vào các thẻ HTML
* DOM Listener: có nhiệm vụ lắng nghe các sự kiện tác động lên thẻ HTML đó
* DOM Navigation dùng để quản lý, thao tác với các thẻ HTML, thể hiện mối quan hệ cha - con của các thẻ HTML
* DOM Node, Nodelist: có nhiệm vụ thao tác với HTML thông qua đối tượng (Object)

Như vậy ta có tổng cộng 8 loại thường được sử dụng như trên, và trong bài này mình không đề cập chi tiết vào nhé các bạn.

**3. Lời kết**

Việc xử lý các thẻ HTML rất là ảo diệu và tuyệt vời cho nên website nào cũng phải sử dụng nó. Trong thực tế các Libraries như jQuery vẫn sử dụng các thể loại DOM này , ví dụ sự kiện click(), hover() thì bản chất nó chính là thuộc nhóm Event + Listener.

**Bài 13: DOM Element trong javascript**

Trong bài này chúng ta sẽ tìm hiểu đến **DOM Element** và sẽ học cách truy xuất đến một thẻ HTML bất kì thông qua các thuộc tính như id, class, name hay thậm chí là tên của thẻ HTML và cú pháp CSS Selector .

Nội dung của bài này chúng ta tìm hiểu các vấn đề sau:

* Tìm thẻ HTML theo ID
* Tìm thẻ HTML theo class
* Tìm thẻ HTML theo tên thẻ
* Tìm thẻ HTML theo selector

**1. Tìm thẻ HTML theo ID**

Để truy xuất tới một thẻ HTML theo ID ta sử dụng cú pháp sau:

|  |
| --- |
| var element = document.getElementById('idname'); |

**Ví dụ**:

|  |
| --- |
| // Lấy thẻ input  var element = document.getElementById('website');  // Lấy giá trị của thẻ input  document.write(element.value); |

**2. Tìm thẻ HTML theo tên của thẻ HTML**

Tên thẻ HTML chính là tên các thẻ như p, a, div, ... Và ta sẽ truy xuất tới nó bằng cú pháp sau:

|  |
| --- |
| var element = document.getElementsByTagName('tagname'); |

**Ví dụ**:

|  |
| --- |
| // Lấy thẻ input  var element = document.getElementsByTagName('input');  // Lấy giá trị của thẻ input  document.write(element[0].value); |

Trong ví dụ này có sự khác biệt đó là đoạn code element[0].value. Tại sao lại như vậy? Như bạn biết trong một trang web có thể có nhiều thẻ HTML giống nhau (ví dụ có hai thẻ div) nên hàm getElementsByTagName() sẽ **trả về một mảng các object** (xem bài mảng trong javascript) chứ không phải là một object nữa, chính vì vậy ta sẽ lấy input thứ nhất nên truyền số 0 vào.

**3. Tìm thẻ HTML theo tên class**

Để tìm các thẻ HTML có class nào đó thì ta dùng cú pháp sau:

|  |  |
| --- | --- |
| 1 | var element = document.getElementsByClassName('input'); |

Tương tự như tìm theo tên thẻ HTML thì tìm theo tên class sẽ trả về một mảng các object nên bạn sẽ phải sử dụng cú pháp truy xuất mảng để chọn đúng đối tượng muốn lấy.

**Ví dụ**: XEM DEMO

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Lấy thẻ input  var element = document.getElementsByClassName('website');    // Lấy giá trị của thẻ input  document.write(element[0].value); |

**4. Tìm thẻ HTML theo cú pháp của Selector CSS**

Khi chọn các thẻ HTML theo class thường sẽ trả về hàng loạt các kết quả, như vậy đôi khi sẽ có những kết quả mà ta không mong đợi. Chính vì vậy javascript DOM có một phương thức kết hợp với [CSS Selector](http://freetuts.net/selector-la-gi-tim-hieu-css-selector-can-ban-345.html) để truy vấn có độ chính xác cao hơn.

Và đây là cú pháp sử dụng:

|  |  |
| --- | --- |
| 1 | var element = document.querySelectorAll("selector.css"); |

**Ví dụ**: Tôi có một đoạn mã HTML như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <html>    <body>      <input type="text" value="thẻ không cần lấy" class="website"/>      <div>          <input type="text" value="Thẻ Cần Lấy" class="website"/>          <input type="text" value="thẻ không cần lấy"/>      </div>    </body>  </html> |

Câu hỏi đặt ra là làm thế nào có thể chọn đúng một thẻ input nằm trong thẻ div và có class="website"?

Trước tiên ta quay lại chút với CSS Selector đã nhé. Trong CSS để chọn thẻ input nằm trong thẻdiv và có class="website" thì cú pháp là:

|  |  |
| --- | --- |
| 1 | div input.website |

Như vậy bài giải sẽ như sau: [XEM DEMO](http://freetuts.net/editor.html?id=287)

|  |  |
| --- | --- |
| 1 | var element = document.querySelectorAll("div input.website"); |

Và có một lưu ý tương tự là kết quả sẽ trả về một mảng các Objects.

**5. Lời kết**

Trên là 4 cách thông thường chúng ta hay sử dụng trong javascript để truy xuất tới một thẻ HTML bất kì. Nếu sau này làm việc với jQuery thì bạn không cần phải nhọc nhằn như thế nữa mà nó có những phương thức ngắn gọn hơn rất nhiều.

**Bài 14: Sự kiện (Event) trong Javascript**

Đáng lẽ bài này mình sẽ tiếp tục giới  thiệu với các bạn [DOM](http://freetuts.net/dom-la-gi-cac-loai-dom-trong-javascript-366.html) nhưng nhận thấy việc học mà không thực hành thì rất nhàm chán và khó nhớ bài, vì vậy trong bài này mình sẽ giới thiệu về**sự kiện trong javascript**. Sau bài này chúng ta có thể làm được nhiều ví dụ hơn và tôi sẽ không mất nhiều thời gian giải thích nữa.

**1. Sự kiện trong javascript là gì?**

Về định nghĩa dưới dạng nghe hiểu thì sự kiện là một hành động nào đó tác động lên đối tượng HTML mà ta có thể bắt được sự kiện này và thực hiện những hành động nào đó.

Mỗi sự kiện chúng ta có thể có nhiều hành động. Ví dụ bạn có một sự kiện là **tết âm lịch** thì bạn sẽ có những hành động trong sự kiện đó là mua đồ mới, dọn dẹp nhà cửa, mua bao lì xì, ..

Xét về mặt thực tế thì ta có dụ thế này: Giả sử bạn xây dựng một form đăng ký tài khoản và bạn muốn bắt sự kiện khi người dùng CLICK vào button đăng ký thì hiện những hành động như validate dữ liệu, thông báo nếu người dùng nhập nội dung không đúng, .. Như vậy bạn cần nhớ rằng mỗi sự kiện chúng ta có thể thực hiện nhiều hành động khác nhau và bao nhiêu hành động thì phụ thuộc vào từng chức năng cụ thể.

**2. Các sự kiện (Events) trong javascrpt**

Sau khái niệm **sự kiện là gì** thì chắc hẳn bạn đang nôn nóng muốn biết **sự kiện trong javascript** là gì rồi nhỉ? Ok ta  bắt đầu ngay luôn.

Có rất nhiều sự kiện và mỗi đối tượng HTML thì lại có những sự kiện khác nhau nên mình chỉ liệt kê được những sự kiện thông dụng hay sử dụng khi các bạn làm việc với javascript để lập trình phía client.

**Bảng các sự kiện thông dụng trong javascript.**

| **STT** | **Event Name** | **Description** |
| --- | --- | --- |
| 1 | onclick | Xảy ra khi click vào thẻ HTML |
| 2 | ondbclick | Xảy ra khi double click vào thẻ HTML |
| 3 | onchange | Xảy ra khi giá trị (value) của thẻ HTML đổi. Thường dùng trong các đối thẻ form input |
| 4 | onmouseover | Xảy ra khi con trỏ chuột bắt đầu đi vào thẻ HTML |
| 5 | onmouseout | Xảy ra khi con trỏ chuột bắt đầu rời khỏi thẻ HTML |
| 6 | onmouseenter | Tương tự như onmouseover |
| 7 | onmouseleave | Tương tự như onmouseout |
| 8 | onmousemove | Xảy ra khi con chuột di chuyển bên trong thẻ HTML |
| 9 | onkeydown | Xảy ra khi gõ một phím bất kì vào ô input |
| 10 | onload | Sảy ra khi thẻ HTML bắt đầu chạy, nó giống như [hàm khởi tạo](http://freetuts.net/ham-khoi-tao-va-ham-huy-trong-lap-trinh-huong-doi-tuong-php-36.html)trong [lập trình hướng đối tượng](http://freetuts.net/tag/lap-trinh-huong-doi-tuong) vậy đó. |
| 11 | onkeyup | Xảy ra khi bạn gõ phím nhưng **lúc bạn nhã phím ra sẽ được kích hoạt** |
| 12 | onkeypress | Xảy ra khi bạn nhấn môt phím vào ô input |
| 14 | onblur | Xảy ra khi con trỏ chuột rời khỏi ô input |
| 15 | oncopy | Xảy ra khi bạn copy nội dung của thẻ |
| 16 | oncut | Xảy ra khi bạn cắt nội dung của thẻ |
| 17 | onpaste | Xảy ra khi bạn dán nội dung vào thẻ |

Ok bây giờ ta sẽ làm một số ví dụ để các bạn thực hành nhé.

**3. Các ví dụ về xử lý sự kiện trong javascript**

**Ví dụ 1**:  Viết chương trình gồm một ô input và một thẻ div dùng để hiển thị nội dung (giá trị của ô input) khi người dùng gõ vào ô input

Vì đề bài yêu cầu khi nhập dữ liệu vào ô input thì hiển thị nội dung bên trong thẻ DIV nên ta có thể sử dụng sự kiện onkeyup. Thứ hai nữa là chúng ta sẽ sử dụng các hàm [DOM Element](http://freetuts.net/dom-element-trong-javascript-367.html)để truy xuất các đối tượng HTML.

**Bài giải**: [XEM DEMO](http://freetuts.net/editor.html?id=288)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <html>    <body>      <script language="javascript">        // Hàm show kết quả        function show\_result()        {          // Lấy hai thẻ HTML          var input = document.getElementById("message");          var div = document.getElementById("result");            // Gán nội dung ô input vào thẻ div          div.innerHTML = input.value;        }      </script>      <input type="text" id="message" value="" onkeyup="show\_result()"/>      <div id="result"></div>    </body>  </html> |

Nếu như bài này bạn sử dụng sự kiện onkeypress hoặc onkeydown thì sẽ có kết quả sai, lý do là những sự kiện này xảy ra khi bạn nhấn phím xuống nên nó sẽ lấy giá trị chưa được cập nhật. Còn sự kiện onkeyup xảy ra khi bạn nhả phím ra nên nó sẽ lấy được giá trị mới.

**Ví dụ 2**: Viết chương trình khi người dùng copy nội dung của website thì thông báo là bạn đã copy thành công

Bài này quá dễ phải không nào. Trong danh sách các sự kiện trên thì có sự kiện oncopy nên ta sẽ sử dụng nó để giải bài này.

**Bài giải**: [XEM DEMO](http://freetuts.net/editor.html?id=289)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <html>    <body>      <script language="javascript">        // Hàm show kết quả        function show\_message()        {          alert("Bạn đã copy thành công");        }      </script>      <h3>Hãy copy dòng chữ dưới đây:</h3>      <div oncopy="show\_message()">Chào mừng các bạn đến với website freetuts.net</div>    </body>  </html> |

**Ví dụ 3:** Viết chương trình tính tổng của hai số nhập vào (tính tự động)

Bài này ta phải tạo 3 ô input và gán sự kiện onkeyup cho 2 ô input đầu tiên, trong sự kiện này sẽ thực hiện tính tổng của hai ô và in kết quả vào ô input thứ 3.

**Bài giải**: [XEM DEMO](http://freetuts.net/editor.html?id=290)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <html>    <body>      <script language="javascript">        // Hàm tính kết quả        function tinh()        {          // Lấy 3 ô input          var a = document.getElementById("a");          var b = document.getElementById("b");          var result = document.getElementById("result");            // Tính tổng hai ô đầu tiên          var tong = parseInt(a.value) + parseInt(b.value);            // Gán giá trị vào ô thứ ba          // Phải kiểm tra tổng hai số này có bị lỗi hay không          if (!isNaN(tong)){            result.value = tong;          }        }      </script>      a: <input type="text" id="a" value="" onkeyup="tinh()"/>      b: <input type="text" id="b" value="" onkeyup="tinh()"/>      Kết quả: <input type="text" id="result" value=""/>    </body>  </html> |

**4. Lời kết**

Làm việc với Javascript thì không thể quên khái niệm event được nên hãy học chắc bài này nhé các bạn. Mình không thể đưa ra hết ví dụ cho các events được nên bạn hãy tự ra bài tập cho mình và tự giải nhé. Riêng nội dung trong bài này mình có sử dụng kiến thúc bài học cũ nên bạn rất dễ theo dõi, nếu quên thì hãy quay lại bài cũ và đọc sư qua rồi thực hành ví dụ. Chúc bạn học javascript vui vẻ nhé!

**Bài 15: Return TRUE/FALSE của Events trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 16-04-2015- View: 8245

**Mục luc:**

* [1. Return true/false của Events trong Javascript](http://freetuts.net/return-true-false-cua-events-trong-javascript-369.html#goto-h2-0)
* [2. Bài tập validate form đơn giản với Event và DOM](http://freetuts.net/return-true-false-cua-events-trong-javascript-369.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/return-true-false-cua-events-trong-javascript-369.html#goto-h2-2)

Trải qua 14 bài học vừa qua thì bạn cũng đã nắm được một số kiến thức khá vững về javascript rồi đấy. Để bổ sung skill cho các bạn thì trong bài này mình sẽ hướng dẫn các bạn cách validate một form căn bản bằng cách sử dụng sự kiện onsubmit của form hoặc là onclick của nút submit, đồng thời mục đích chính là hướng dẫn bạn cách phân biệt **return trong events**.

Vì chúng ta chưa học đến các kỹ thuật cao như [Regular Expression](http://freetuts.net/tag/regular-expression) nên trong bài này mình chỉ thực hành validate đơn giản, mục đích là hướng dẫn các bạn các bước validate và cách sử dụng lệnh return đúng nhất.

**1. Return true/false của Events trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Có lẽ bạn thắc mắc tại sao lại nêu ra vấn đề return true/false này nhỉ ?. Đây là một vấn đề khá quan trọng đấy nếu bạn sử dụng không đúng thì bạn sẽ không biết tại sao mình code sai và không biết chỗ nào để sửa cả.

Trong một sự kiện sẽ có hai trạng thái là hành động đúng (TRUE) và hành động sai (FALSE) và để thể hiện hai trạng thái này thì ta sử dụng cú pháp return TRUE/FAlSE trong function của event.

Có hai cách return thông dụng nhất như sau:

* **Cách 1**: Return tại đoạn code event trong HTML luôn, ví dụ:

|  |  |
| --- | --- |
| 1 | <input type="text" onkeypress="return false" /> |

* **Cách 2**: Tạo một hàm xử lý sự kiện, lúc này hàm này phải return và trong HTML bạn cũng phải return, ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <script language="javascript">  function check()  {      return false;  }  </script>  <input type="text" onkeypress="return check()" /> |

Bạn phải thực sự hiểu ý nghĩa của từng loại [sự kiện trong javascript](http://freetuts.net/su-kien-event-trong-javascript-368.html) thì mới áp dụng đúng được. ví dụ mình cần viết chương trình không cho người dùng nhập bất kỳ chữ gì vào một ô input thì nếu bạn sử dụng sự kiện onkeyup sẽ sai, lý do là sự kiện này sảy ra khi bạn nhã phím nên bạn nhập rồi nó mới kích hoạt. Thay vì vậy thì bạn sử dụng sự kiện onkeypress sẽ thành công.

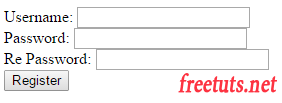
**2. Bài tập validate form đơn giản với Event và DOM**

Trong bài tập này ta sẽ sử dụng sự kiện onclick của nút submit, đồng thời sử dụng [DOM](http://freetuts.net/dom-la-gi-cac-loai-dom-trong-javascript-366.html) để lấy các giá trị trong form.

Giả sử ta có đoạn HTML sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <!DOCTYPE html>  <html>      <body>        <form method="post" action="">          Username: <input type="text" value="" name="username" id="username"/> <br/>          Password: <input type="text" value="" name="password" id="password"/> <br/>          Re Password: <input type="text" value="" name="re-password" id="re-password"/> <br/>          <input type="submit" value="Register" />        </form>      </body>  </html> |

Và đây là giao diện khi chạy lên:



**Yêu cầu của bài tập như sau**:

* Tên đăng nhập không được để trống
* Mật khẩu không được để trống
* Mật khẩu nhập lại phải giống với mật khẩu đã nhập ở trên

**Hướng giải**:

* Tạo một javascript function validate()
* Sử dụng return validate() trong sự kiện onclick của button Register
* Trong hàm validate() sử dụng **DOM** để lấy giá trị các ô input, sau đó kiểm tra các giá trị và return về true/false

**Bài giải**: [XEM DEMO](http://freetuts.net/editor.html?id=291)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <!DOCTYPE html>  <html>      <body>          <script language="javascript">            function validate()            {                 // Lấy giá trị               var username = document.getElementById("username").value;               var password = document.getElementById("password").value;               var re\_password = document.getElementById("re-password").value;                  // Kiểm tra các giá trị              if (username == ""){                alert("Bạn chưa nhập tên đăng nhập");                return false;              }                if (password == ""){                alert("Bạn chưa nhập mật khẩu");                return false;              }                  if (password != re\_password){                alert("Mật khẩu nhập lại không đúng");                return false;              }                return true;            }          </script>          <form method="post" action="">              Username: <input type="text" value="" name="username" id="username"/> <br/>              Password: <input type="text" value="" name="password" id="password"/> <br/>              Re Password: <input type="text" value="" name="re-password" id="re-password"/> <br/>              <input type="submit" value="Register" onclick="return validate()" />          </form>      </body>  </html> |

Bạn chú ý trong button mình có thêm một sự kiện onclick đấy nhé.

**3. Lời kết**

Tóm lại **return của events trong javascript** dùng để xác nhận một sự kiện nào đó thực thi thành công hay thất bại, từ đó đối tượng HTML sẽ có những xử lý **cho phép** hoặc **không cho phép** thao tác đó thực hiện. Thông thường chúng ta hay vận dụng nó trong các bài toán validate form hoặc những bài toán liên quan đến sự kiện trong javascript.

**Bài 16: DOM HTML trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 16-04-2015- View: 11232

**Mục luc:**

* [1. Thay đổi và lấy nội dung bên trong thẻ HTML](http://freetuts.net/dom-html-trong-javascript-370.html#goto-h2-0)
* [2. Thay đổi và lấy giá trị thuộc tính thẻ HTML bằng Javascript](http://freetuts.net/dom-html-trong-javascript-370.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/dom-html-trong-javascript-370.html#goto-h2-2)

Chúng ta đã được học [DOM Elements trong javascript](http://freetuts.net/dom-element-trong-javascript-367.html) rồi nhỉ, vậy thì trong bài này chúng ta tiếp tục tìm hiểu một nhóm khác đó là DOM HTML chuyên xử lý các vấn đề liên quan đến nội dung, thuộc tính của thẻ HTML.

Đáng lẽ mình sẽ trình bày xuyên suốt DOM luôn nhưng nhận thấy việc làm ví dụ mà không kết hợp với [sự kiện trong javascript](http://freetuts.net/su-kien-event-trong-javascript-368.html) thì rất khó học cho nên mình đã chèn hai bài vào rồi mới trình bày các thành phần tiếp theo của DOM .

Say đây là nội dung chúng ta sẽ được học trong bài này:

* Cách thay đổi, lấy nội dung của thẻ HTML
* Cách thay đổi, lấy nội dung của thuộc tính HTML

**1. Thay đổi và lấy nội dung bên trong thẻ HTML**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Để lấy nội dung bên trong một thẻ HTML thì chúng ta sử dụng cú pháp như sau:

|  |  |
| --- | --- |
| 1 | var html = document.getElementById("content").innerHTML |

Và để thay đổi nội dung cho một thẻ HTML thì ta dùng cú pháp sau:

|  |  |
| --- | --- |
| 1 | var html = document.getElementById("content").innerHTML = "<h1>Nội dung</h1>"; |

**Ví dụ**: Trong ví dụ này ta sẽ viết chương trình thay đổi nội dung HTML của một thẻ DIV và lấy nội dung bên trong của một thẻ DIV

[XEM DEMO](http://freetuts.net/editor.html?id=292)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <html>    <body>      <script language="javascript">        // Hàm thiết lập nội dung cho thẻ div#content        function set\_content()        {          document.getElementById("content").innerHTML = "<h1>Nội dung đã được thay đổi</h1>";        }          // Hàm lấy nội dung cho thẻ div#content        function get\_content()        {          var html = document.getElementById("content").innerHTML;          alert("Nội dung cần lấy là: " + html);        }        </script>      <div id="content">Nội dung của thẻ DIV</div>      <input type="button" value="Lấy nội dung" id="get\_content" onclick="get\_content()" />      <input type="button" value="Thay đổi nội dung" id="set\_content" onclick="set\_content()" />    </body>  </html> |

**2. Thay đổi và lấy giá trị thuộc tính thẻ HTML bằng Javascript**

Để thay đổi giá trị của một thuộc tính HTML bất kì thì ta sử dụng cú pháp sau:

|  |  |
| --- | --- |
| 1 | document.getElementById("element").attributeName = "new value"; |

Để lấy giá trị của một thuộc tính HTML ta sử dụng cú pháp sau:

|  |  |
| --- | --- |
| 1 | var value = document.getElementById("element").attributeName; |

Quá đơn giản phải không nào, rất giống với cách thay đổi và lấy nội dung bên trong thẻ HTML. Từ đây có thể suy ra rằng trong Javascript để thiết lập (set) và lấy (get) thì sử dụng chung một cú pháp, chỉ khác nhau ở chỗ **gán bằng** và **không có gán bằng**.

**Ví dụ**: Xây dựng chương trình khi click vào một button thì chuyển nó thành textbox, và tiếp tục click vào textbox thì sẽ đổi thành button

[XEM DEMO](http://freetuts.net/editor.html?id=293)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | <html>      <body>          <script language="javascript">            function change()            {               // Lấy đối tượng               var object = document.getElementById("object");                 // lấy thuộc tính type               var type = object.type;                 // kiểm tra thuộc tính type và thay đổi              if (type == "button"){                   object.type = 'text';              }              else{                  object.type = "button";              }              }          </script>          <input type="button" value="CLick me" onclick="change()" id="object" />      </body>  </html> |

**3. Lời kết**

Như vậy là bạn đã biết hai cách xử lý thuộc tính và nội dung của thẻ HTML bằng javascript rất thông dụng rồi đấy, thực tế bạn có thể sử dụng những mánh khóe để xử lý bài toán một cách tốt hơn. Ví dụ bạn cần bổ sung một đoạn text vào trong một thẻ HTML thì sẽ không có hàm nào bổ trợ, tuy nhiên bạn có thể sử dụng một mẹo nhỏ thế này:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var content\_append = 'nội dung cần thêm vào';    // Lấy đối tượng  var object = document.getElementById("object");    // Lấy nội dung hiện tại  var content\_current = object.innerHTML;    // Bổ sung nội dung vào đối tượng  object.innerHTML = content\_current + content\_current; |

Đấy, cho nên việc lập trình là vô vàn hướng giải và việc lựa chọn một hướng đi đúng mới là vấn đề. Và qua bài này bạn thấy sức mạnh của [DOM](http://freetuts.net/dom-la-gi-cac-loai-dom-trong-javascript-366.html) rồi phải không nào

**Bài 17: DOM CSS trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 17-04-2015- View: 6924

**Mục luc:**

* [1. Thay đổi CSS bằng Javascript](http://freetuts.net/dom-css-trong-javascript-371.html#goto-h2-0)
* [2. Ví dụ thay đổi CSS bằng Javascript](http://freetuts.net/dom-css-trong-javascript-371.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/dom-css-trong-javascript-371.html#goto-h2-2)

Tiếp tục serie học CSS căn bản thì trong bài thứ 17 này chúng ta vẫn tiếp tục tìm hiểu thêm một thành phần của DOM nữa đó là DOM dùng để xử lý CSS. Chúng ta sẽ tìm hiểu các thao tác**thay đổi CSS bằng javascript**, xử lý các hiệu ứng để làm website động ở client (DHTML).

**1. Thay đổi CSS bằng Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Style bản chất nó cũng là một thuộc tính của các thẻ HTML nhưng bạn không thể sử dụng [DOM HTML](http://freetuts.net/dom-html-trong-javascript-370.html) để thiết lập hay xóa bỏ CSS được mà phải thông qua một đối tượng biệt khác đó là style.

Đối tượng style này sẽ chứa tất cả các thuộc tính của CSS và chúng ta sẽ dễ dàng thao tác với chúng bằng cú pháp riêng, và như thường lệ chúng ta có hai thao tác chính đó là **thiết lập CSS** và **lấy giá trị CSS** hiện tại.

Cú pháp thiết lập CSS bằng Javascript:

|  |  |
| --- | --- |
| 1 | document.getElementById("object").style.cssName = 'something'; |

Cú pháp lấy giá trị CSS bằng Javascript:

|  |  |
| --- | --- |
| 1 | var value = document.getElementById("object").style.cssName; |

Trường hợp thuộc tính có dấu gạch ngang như: font-size, line-height, margin-bottom thì thuộc tính đó trong style sẽ có tên là fontSize, lineHeight, marginBottom ,nghĩa là sẽ bỏ đi dấu gạch ngang và viết hoa ký tự đầu tiên của chữ thứ hai.

|  |  |
| --- | --- |
| 1  2  3  4 | document.getElementById("object").style.fontSize = 'something';  document.getElementById("object").style.lineHeight = 'something';  document.getElementById("object").style.marginBottom = 'something';<br />  <br /> |

Lưu ý rằng có những thuộc tính nếu bạn chưa thiết lập CSS cho nó thì khi bạn lấy giá trị sẽ là một giá trị rỗng.

**2. Ví dụ thay đổi CSS bằng Javascript**

Để rõ ràng và dễ hiểu thì chúng ta làm một ví dụ:

**Ví dụ 1**: Viết một chương trình gồm 4 buttons và 1 thẻ div, khi click vào từng button thì sẽ thiết lập **màu sắc**, **background**, **chiều cao**, **font size** của thẻ div.

Để làm bài này thì chúng ta phải sử dụng [sự kiện trong javascript](http://freetuts.net/su-kien-event-trong-javascript-368.html) đó là onclick, mỗi khi click vào mỗi button sẽ giải quyết một vấn đề của bài toán.

Các bước thực hiện:

* Tạo 4 functions thực hiện 4 nhiệm vụ như đề bài yêu cầu
* Gán mỗi function vào sự kiện onclick của mỗi button

**Bài giải**: [XEM DEMO](http://freetuts.net/editor.html?id=294)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | <html>      <body>          <script language="javascript">              function change\_background()              {                 document.getElementById("message").style.background = 'red';              }                function change\_color()              {                 document.getElementById("message").style.color = 'blue';              }                function change\_height()              {                 document.getElementById("message").style.height = '500px';              }                function change\_font\_size()              {                 document.getElementById("message").style.fontSize = '500px';              }          </script>          <div id="message">            Chào mừng các bạn đến với freetuts.net          </div>          <input type="button" value="Change backgroud" onclick="change\_background()"/>          <input type="button" value="Change color" onclick="change\_color()"/>          <input type="button" value="Change height" onclick="change\_height()"/>          <input type="button" value="Change fontsize" onclick="change\_font\_size()"/>      </body>  </html> |

**Ví dụ 2**: Viết chương trình đăng nhập và validate thông tin username, password. Nếu người dùng không nhập username hoặc mật khẩu thì hiển thị thông báo message trong một thẻ div và chữ màu đỏ, ngược lại thì thông báo validate thành công và chữ màu xanh

Với bài này bạn cần phải sử dụng thêm [DOM Element](http://freetuts.net/dom-element-trong-javascript-367.html) để lấy giá trị của các ô input, đồng thời sử dụng DOM HTML để thay đổi nội dung của message và sử dụng **DOM CSS** để thay đổi màu sắc.

**Bài giải**: [XEM DEMO](http://freetuts.net/editor.html?id=295)

|  |
| --- |
| <html>      <body>          <script language="javascript">            function validate() {              // Lấy giá trị input               var username = document.getElementById("username").value;               var password = document.getElementById("password").value;                // Lấy đối tượng message              var message = document.getElementById("message");                // Validate              if (username == "" || password == "") {                  message.innerHTML = "Bạn chưa nhập đầy đủ thông tin!";                  message.style.color = "red";              } else{                  message.innerHTML = "Chúc mừng ban, validate thành công!";                  message.style.color = "blue";              }          }          </script>          Username: <input type="text" value="" id="username" /> <br/>          Username: <input type="password" value="" id="password" /> <br/>          <div id="message"></div>          <input type="button" value="Login" onclick="validate()"/>      </body>  </html> |

**3. Lời kết**

Những bài toán dạng như thế này thì chúng ta hay gọi là DHTML, nghĩa là HTML động có các hiệu ứng thay đổi giao diện bằng cách sử dụng Javascript kết hợp với giá trị CSS của HTML. Tất cả các thuộc tính của CSS được lưu trong đối tượng style (có thể gọi là thuộc tính style) của đối tượng HTML nên bạn tránh ghi nhầm bỏ đi chữ style nhé.

Có một điểm lưu ý trong bài này mình muốn nhắc lại trước khi nhảy qua bài tiếp là tên của thuộc tính nếu có dấu gạch ngang thì bạn phải bỏ dấu gạch ngang, đồng thời viết hoa từ chữ thứ hai nhé, ví dụ fontSize, lineHeight, ... Còn những thuộc tính khác thì tên của nó viết bình thường

**Bài 18: Thêm sự kiện (Event) bằng Javascript**

Trong bài tìm hiểu [sự kiện trong javascript](http://freetuts.net/su-kien-event-trong-javascript-368.html) chúng ta đã tìm hiểu cách gán một sự kiện và xử lý sự kiện đó.  Tuy nhiên chúng ta vẫn còn một cách khác khá hay và nó rất giống với jQuery đó là sử dụng code javascript để thiết lập một sự kiện cho thẻ HTML.

Nhung trước khi vào vấn đề chính chúng ta sẽ tìm hiểu một lý do tại sao chúng ta nên **thêm sự kiện bằng Javascript** thay vì gán trực tiếp nó trong code HTML đã nhé.

**1. Tình huống thêm sự kiện trong HTML không tốt**

Giả sử mình có một đoạn mã HTML như sau:

|  |
| --- |
| <ul>    <li><a href="#" onclick="do\_something()">Tin công nghệ</a></li>    <li><a href="#" onclick="do\_something()">Tin trong nước</a></li>    <li><a href="#" onclick="do\_something()">Tin thời sự</a></li>  </ul> |

Trong đoạn code này mình đã gán một hàm tên là do\_something() vào các thẻ a. Như vậy giả sử ta có 100 thẻ a như vậy thì ta phải gán 100 lần, sau một ngày nào đó bạn sửa lại tên của function từ do\_something() sang do\_validate() thì bắt buộc bạn phải replace 100 lần. Điều này là một cơn ác mộng phải không nào, chưa tính đến vụ cấu trúc như vậy sẽ không tối ưu [SEO Onpage](http://freetuts.net/tag/seo-onpage) nữa.

Để giải quyết vấn đề này thì chúng ta sẽ **sử dụng Javascript để thêm sự kiện** cho các thẻ HTML đó.

**2. Thêm sự kiện cho một thẻ HTML bằng javascript**

Để thêm sự kiện bằng Javascript thì bạn sẽ sử dụng cú pháp như sau:

|  |
| --- |
| elementObject.eventName  = function(){      // do something  }; |

**Trong đó:**

* **elementObject** là đối tượng HTML mà chúng ta sử dụng [DOM](http://freetuts.net/dom-la-gi-cac-loai-dom-trong-javascript-366.html) để lấy.
* **eventName**là tên của event như onclick, onchange, ...

**Ví dụ**: Thêm sự kiện click cho button có id="show-btn"

|  |
| --- |
| <html>      <body>          <input type="button" id="show-btn" value="Click me" />            <script language="javascript">              // Lấy đối tượng              var button = document.getElementById("show-btn");                // Thêm sự kiện cho đối tượng              button.onclick = function()              {                alert("Bạn vừa click vào button");              };          </script>      </body>  </html> |

**3. Thêm sự kiện cho nhiều thẻ HTML bằng Javascrip**

Ở phần 2 là chúng ta sử dụng phương thức document.getElementById() nên kết quả nó sẽ trả về chỉ là **một đối tượng**nên có thể gán trược tiếp button.onclick = function(){}. Nhưng nếu bạn sử dụng truy vấn **DOM theo selector** thì kết quả nó sẽ trả về **một mảng các đối tượng**, lúc này ban phải sử dụng vòng lặp để thiết lập sự kiện.

**Cú pháp như sau**:

|  |
| --- |
| // Lấy đối tượng html, bạn có thể sử dụng các DOM Element khác để lấy  var elementObjs = document.getElementsByTagName('element');  // Lặp qua từng phần tử trong kết quả và gán sự kiện  for (var i = 0; i < elementObjs.length; i++) {      elementObjs[i].eventName = function() {         // Do simething      };  } |

**Ví dụ**: Thêm sự kiện khi click vào tất cả các thẻ a có class="show" thì thông báo lên câu chào.

|  |
| --- |
| <ul>      <li><a href="#" class="show">Tin công nghệ</a></li>      <li><a href="#">Tin trong nước</a></li>      <li><a href="#" class="show">Tin thời sự</a></li>  </ul> |

Như vậy trong đề bài này muốn khi click vào thẻ **a đầu tiên** và **thẻ a thứ 3** thì sẽ thông báo câu chào.

**Bài giải**:

|  |
| --- |
| <html>      <body>          <ul>              <li><a href="#" class="show">Tin công nghệ</a></li>              <li><a href="#">Tin trong nước</a></li>              <li><a href="#" class="show">Tin thời sự</a></li>          </ul>          <script language="javascript">              // Lấy đối tượng              var a\_list = document.getElementsByClassName("show");                // Lặp và gán sự kiện              for (var i = 0; i < a\_list.length; i++){                  a\_list[i].onclick = function() {                      alert('Xin chào, bạn vừa click vào tôi');                      // return false để khỏi reload trang                      return false                  };              }          </script>      </body>  </html> |

Trong ví dụ này bạn lưu ý mình có return false ở phần thực hiện sự kiện vì mình muốn khi click vào thì thẻ a thì không thực hiện chuyển hướng (xem lại bài [return true/false của event](http://freetuts.net/return-true-false-cua-events-trong-javascript-369.html)).

**4. Lời kết**

Tóm lại trong bài bạn cần hiểu lý do tại sao nên **gán sự kiện bằng Javascript** thay vì gán trực tiếp bằng mã HTML, đồng thời hiểu và phân biệt được khi nào thì dùng vòng lặp để gán sự kiện và khi nào có thể gán trực tiếp mà không cần sử dụng vòng lặp.

* Nếu  kết quả [DOM Element](http://freetuts.net/dom-element-trong-javascript-367.html) trả về một object thì ta gán sự kiện trực tiếp.
* Nếu kết quả DOM Element trả về một mảng các object thì ta phải sử dụng vòng lặp để lặp qua từng đối tượng để gán sự kiện.

Các ví dụ trong bài mình làm khá đơn giản mục đích giúp bạn dễ học hơn, sau serie này thì sẽ có một serie bài tập javascript để các bạn ôn lại kiến thức. Hy vọng đây là một serie có ích cho các bạn.

**Bài 19: Hàm addEventListener() trong Javascript**

Bài trước chúng ta đã học cách sử dụng Javascript để thêm một sự kiện vào một đối tượng HTML nhưng chúng ta sử dụng cú pháp gán bình thường. Vậy trong bài này chúng ta tìm hiểu một cách cao siêu hơn đó là sử dụng hàm **addEventListener()**để thêm một sự kiện vào đói tượng HTML.

**1. Hàm addEventListener() trong Javascript**

Để thêm sự kiện cho đối tượng HTML thì chúng ta có cú pháp như sau (xem lại bài [thêm sự kiện bằng javascript](http://freetuts.net/them-su-kien-event-bang-javascript-373.html)):

|  |  |
| --- | --- |
| 1  2  3 | elementObject.eventName  = function(){      // do something  }; |

Nhưng nếu bạn sử dụng hàm addEventListener() thì cú pháp như sau:

|  |  |
| --- | --- |
| 1  2  3 | elementObject.addEventListener('eventName', function(e){      // do something  }); |

**Trong đó**:

* **eventName** là tên của sự kiện bỏ đi chữ on, ví dụ click, change, ...
* function ở tham số thứ hai chính là hàm sẽ được chạy khi sự kiện **eventName** được kích hoạt

**Ví dụ 1**: Xây dựng chức năng khi nhập dữ liệu vào ô input thì hiển thị giá trị của ô input đó ra bên ngoài

Với bài này ta sử dụng sự kiện onkeyup và bỏ đi chữ on sẽ là keyup.

**Bà giải**: [XEM DEMO](http://freetuts.net/editor.html?id=299)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <html>      <body>          <input type="text" id="txt-val" value="" />          <div id="result"></div>          <script language="javascript">              // Lấy đối tượng              var input = document.getElementById("txt-val");                // Thêm sự kiện cho đối tượng              input.addEventListener('keyup', function(){                  // Gán giá trị vào div                  document.getElementById('result').innerHTML = input.value;              });          </script>      </body>  </html> |

Như ta biết mỗi sự kiện có thể có nhiều hành động nên bạn có thể bổ sung hành động cho một sự kiện và cú pháp tương tự như trên.

**Ví dụ 2**: Bổ sung thêm chức năng ở ví dụ 1 như sau: nếu chiều dài của chuỗi nhập vào nhiều hơn 5 ký tự thì thông báo

**Bài giải**: [XEM DEMO](http://freetuts.net/editor.html?id=300)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <html>      <body>          <input type="text" id="txt-val" value="" />          <div id="result"></div>          <script language="javascript">              // Lấy đối tượng              var input = document.getElementById("txt-val");                // Thêm sự kiện cho đối tượng              input.addEventListener('keyup', function(){                  // Gán giá trị vào div                  document.getElementById('result').innerHTML = input.value;              });                // Bổ sung hành động nữa              input.addEventListener('keyup', function(){                  if (input.value.length > 5){                      alert("Bạn đã nhập nhiều hơn 5 ký tự");                  }              });          </script>      </body>  </html> |

**2. Thêm sự kiện cho đối tượng window**

Đối tượng window ta có thể ví nó như cửa sổ trình duyệt browser nên nó sẽ có một số sự kiện riêng, điển hình là sự kiện resize browser. Như vậy với hàm addEventListener() ta cũng có thể thêm sự kiện cho window.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=301)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <html>      <body>          <h4>Bạn hãy zoom trình duyệt</h4>          <div id="result"></div>          <script language="javascript">            window.addEventListener("resize", function(){              document.getElementById("result").innerHTML = "Bạn vừa zoom Browser";            });          </script>      </body>  </html> |

**3. Truyền tham số vào Event trong hàm addEventListener()**

Nếu bạn muốn truyền tham số vào thì bắt buộc bạn phải tạo một hàm khác rồi gọi nó từ hàm addEventListener().

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=302)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | // Lấy đối tượng  var button = document.getElementById("btn");    // Thêm sự kiện cho đối tượng  button.addEventListener('click', function(){      do\_something(2, 3);  });        function do\_something(a, b)  {    alert( a + b);  } |

**4. Lời kết**

Thực tế bạn có thể sử dụng cách thêm sự kiện đã học ở bài trước nhưng sẽ không chuyên nghiệp và hay bằng cách sử dụng hàm addEventListener(). Từ hàm này bạn có thể liên tưởng qua Events trong jQuery có cách xử lý y chang như vậy nên sau này nếu đi phỏng vấn có hỏi câu này thì bạn trả lời bản chất xử lý sự kiện trong jQuery cũng sử dụng hàm này nhé.

**Bài 20: Đối tượng this trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 18-04-2015- View: 4898

**Mục luc:**

* [1. This trong Javascript là gì?](http://freetuts.net/doi-tuong-this-trong-javascript-376.html#goto-h2-0)
* [2. Lời kết](http://freetuts.net/doi-tuong-this-trong-javascript-376.html#goto-h2-1)

Có nhiều bạn gửi email hỏi mình rằng đối tượng **this trong javascript** có ý nghĩa như thế nào mà ở một số đoạn code trên mạng có sử dụng. Thì thực ra bản chất nó giống như cái tên của nó là ám chỉ đối tượng hiện tại (this). Vấn đề này rất giống như trong [lập trình hướng đối tượng](http://freetuts.net/tag/lap-trinh-huong-doi-tuong) khi muốn truy xuất đến các phương thức hay thuộc tính thì phải sử dụng $this->name.

Ok để dễ hiểu hơn thì chúng ta sẽ tìm hiểu một số ví dụ về đối tượng this.

**1. This trong Javascript là gì?**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Đối tượng this chính là đối tượng hiện tại đang được sử dụng hoặc đang truy cập tới. Ví dụ khi bạn viết một hành động cho [sự kiện](http://freetuts.net/su-kien-event-trong-javascript-368.html) click thì nếu trong hành động đó có sử dụng this lúc này this chính là thẻ HTML mà ta click.

**Ví dụ**: Đối tượng this trong event

[XEM DEMO](http://freetuts.net/editor.html?id=303)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // Lấy đối tượng  var button = document.getElementById('btn');    // Gán sự kiện  button.addEventListener("click", function(){  // Lấy thuộc tính type của đối tượng đang xử lý  // chính là button có id="btn"      alert(this.type);  }); |

Như vậy trong ví dụ này this chính là thẻ HTML mà chúng ta đang xử lý và đó chính là thẻ HTML có id="btn", vì vậy bạn có thể sử dụng [DOM](http://freetuts.net/dom-la-gi-cac-loai-dom-trong-javascript-366.html) để lấy các giá trị thuộc tính hoặc gán thêm sự kiện khác vẫn được.

**Ví dụ**: Truyền đối tượng this trong HTML

[XEM DEMO](http://freetuts.net/editor.html?id=304)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <script language="javascript">      function show\_type(obj)      {          alert(obj.type);      }  </script>  <input type="button" onclick="show\_type(this)" value="Check" /> |

Trong ví dụ này chúng ta truyền this ở dạng một tham số nên ở hàm xử lý show\_type() tuy ta đặt tên tham số là obj nhưng bản chất nó chính là đối tượng this.

**Tóm lại**: qua hai ví dụ này bạn có thể ví các thẻ HTML như là một đối tượng nên nó sẽ có các thuộc tính và các phương thức (event), lúc này this chính là đối tượng HTML đó nên ta có thể  sử dụng nó để gọi đến các thuộc tính và phương thức.

**2. Lời kết**

Vẫn còn một số ví dụ liên quan đến this nữa nhưng nó liên quan tới **Javascript nâng cao** nên mình sẽ không trình bày trong bài này mà ở các bài nâng cao mình sẽ có sử dụng và giải thích qua. Cái quan trọng chính là hiểu được **this là gì** và cách sử dụng nó khi kêt hợp với [DOM HTML](http://freetuts.net/dom-html-trong-javascript-370.html).

Bài này cũng tương đối ngắn vì nội dung cũng không có gì để trình bày, cái quan trọng nhất là bạn cần nắm được then chốt của vấn đề đó là **cách sử dụng đối tượng this** để sau này gặp bài tập hoặc ví dụ thì bạn sẽ dễ dàng hiểu bài hơn.

**Bài 21: Bài tập javascript - xây dựng menu dropdow**

Đăng bởi: TheHalfHeart- Vào ngày: 18-04-2015- View: 6218

**Mục luc:**

* [1. Xây dựng HTML - CSS cho dropdown menu](http://freetuts.net/bai-tap-javascript-xay-dung-menu-dropdow-372.html#goto-h2-0)
* [2. Code Javascript hiệu ứng Dropdown menu](http://freetuts.net/bai-tap-javascript-xay-dung-menu-dropdow-372.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/bai-tap-javascript-xay-dung-menu-dropdow-372.html#goto-h2-2)

Trong bài này chúng ta sẽ thực hành xây dựng một **menu dropdown bằng javascript** chứ không sử dụng jquery. Đây là một dạng menu khá thông dụng và có một plugin trên mạng chuyên dùng để tạo đó là **plugin accordion jquery**. Tuy nhiên chúng ta đang học nên bắt buộc phải sử dụng Javascript nhé các bạn.

**1. Xây dựng HTML - CSS cho dropdown menu**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Trước tiên chúng ta cần xây dựng HTML cho menu. Bạn tạo một file bất kì với đuôi là .html, sau đó viết một đoạn mã HTML xây dựng menu như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <ul id="dropdown">      <li><a href="#">Trang chủ</a></li>      <li>          <a href="#">Sản phẩm</a>          <ul>              <li><a href="#">Gạch cao cấp</a></li>              <li><a href="#">Gạch thủ công</a></li>              <li><a href="#">Gạch hoa văn</a></li>          </ul>      </li>      <li>          <a href="#">Tin tức</a>          <ul>              <li><a href="#">Tin công nghệ</a></li>              <li><a href="#">Tin trong nước</a></li>              <li><a href="#">Tin thời sự</a></li>          </ul>      </li>      <li>          <a href="#">Dịch vụ</a>          <ul>              <li><a href="#">Thiết kế web</a></li>              <li><a href="#">Quảng cáo web</a></li>              <li><a href="#">Thiết kế đồ họa</a></li>          </ul>      </li>  </ul> |

Ok, tiếp theo chúng ta sẽ viết CSS style cho menu này:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #dropdown {    width: 200px  }  #dropdown li{    list-style:none;    line-height: 30px;    background: blue;    margin: 2px;  }    #dropdown li a{    margin-left: 10px;    color: #FFF;    text-decoration: none;  }    #dropdown ul{    display:none;    padding: 0px;    background: pink  }    #dropdown ul li{    background-color: pink  } |

Và đây là giao diện sau khi chúng ta thực hiện xong bước 1.

[XEM DEMO](http://freetuts.net/editor.html?id=296)



**2. Code Javascript hiệu ứng Dropdown menu**

Nhiệm vụ tiếp theo là cần viết một đoạn mã javascript để thực hiện hiệu ứng khi click vào các menu thì sẽ xổ ra các menu con. Lúc này bắt buộc chung ta phải sử dụng hàm[addEventListener() trong javascript](http://freetuts.net/ham-addeventlistener-trong-javascript-374.html) để gán sự kiện click cho các thẻ a của menu. Tuy nhiên có một lưu ý rằng vì chúng ta **gán cho nhiều menu** nên khi gán sự kiện vào chúng ta phải thực hiện [vòng lặp for](http://freetuts.net/vong-lap-for-trong-javascript-270.html). Vấn đề này mình đã trình bài ở các bài trước rồi nên sẽ không nhắc lại nữa.

Và đây là đoạn mã Javascript code menu dropdow như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | // Lấy danh sách menu  var menu = document.querySelectorAll('#dropdown > li');    // Lặp qua từng menu để gán sự kiện click  for (var i = 0; i < menu.length; i++)  {      menu[i].addEventListener("click", function()                  {          // Ẩn hết menu con          var menuList = document.querySelectorAll('#dropdown > li > ul');          for (var j = 0; j < menuList.length; j++){                                  menuList[j].style.display = "none";          }            // Hiển thị menu hiện tại          // đối tượng this chính là thẻ li hiện tại          // nên ta sử dụng mảng childrent để lấy danh sách thẻ con          // mà thẻ ul nằm ở vị trí thứ 2 nên trong mảng con nó          // sẽ có vị trí là 1 (mảng bắt đầu từ 0)          this.children[1].style.display = "block";      });  } |

Và  cuối cùng là full code cho ví dụ này:

[XEM DEMO](http://freetuts.net/editor.html?id=305)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87 | <html>      <head>          <style type="text/css">              #dropdown {                  width: 200px              }              #dropdown li{                  list-style:none;                  line-height: 30px;                  background: blue;                  margin: 2px;              }                #dropdown li a{                  margin-left: 10px;                  color: #FFF;                  text-decoration: none;              }                #dropdown ul{                  display:none;                  padding: 0px;                  background: pink              }                #dropdown ul li{                  background-color: pink              }            </style>      </head>        <body>          <ul id="dropdown">              <li><a href="#">Trang chủ</a></li>              <li>                  <a href="#">Sản phẩm</a>                  <ul>                      <li><a href="#">Gạch cao cấp</a></li>                      <li><a href="#">Gạch thủ công</a></li>                      <li><a href="#">Gạch hoa văn</a></li>                  </ul>              </li>              <li>                  <a href="#">Tin tức</a>                  <ul>                      <li><a href="#">Tin công nghệ</a></li>                      <li><a href="#">Tin trong nước</a></li>                      <li><a href="#">Tin thời sự</a></li>                  </ul>              </li>              <li>                  <a href="#">Dịch vụ</a>                  <ul>                      <li><a href="#">Thiết kế web</a></li>                      <li><a href="#">Quảng cáo web</a></li>                      <li><a href="#">Thiết kế đồ họa</a></li>                  </ul>              </li>          </ul>          <script language="javascript">                // Lấy danh sách menu              var menu = document.querySelectorAll('#dropdown > li');                // Lặp qua từng menu để gán sự kiện click              for (var i = 0; i < menu.length; i++)              {                  menu[i].addEventListener("click", function()                  {                      // Ẩn hết menu con                      var menuList = document.querySelectorAll('#dropdown > li > ul');                      for (var j = 0; j < menuList.length; j++) {                          menuList[j].style.display = "none";                      }                        // Hiển thị menu hiện tại                      // đối tượng this chính là thẻ li hiện tại                      // nên ta sử dụng mảng childrent để lấy danh sách thẻ con                      // mà thẻ ul nằm ở vị trí thứ 2 nên trong mảng con nó                      // sẽ có vị trí là 1 (mảng bắt đầu từ 0)                      this.children[1].style.display = "block";                  });              }          </script>      </body>  </html> |

**3. Lời kết**

Ví dụ này hết sức đơn giản nhưng có sử dụng nhiều kiến thức mà ta đã học **như đối tượng**[**this**](http://freetuts.net/doi-tuong-this-trong-javascript-376.html), hàm thêm [sự kiện](http://freetuts.net/su-kien-event-trong-javascript-368.html) addEventListener(), cách sử dụng vòng lặp và [DOM](http://freetuts.net/dom-la-gi-cac-loai-dom-trong-javascript-366.html) để gán sự kiện, ... Sau này khi bạn học xong Javascript thì các bạn học jQuery sẽ có rất nhiều hàm bổ trợ giải quyêt vấn đề này một cách nhanh nhất.

**Bài 22: Hàm removeEventListener() trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 19-04-2015- View: 3233

**Mục luc:**

* [1. Hàm removeEventListener() trong Javascript](http://freetuts.net/ham-removeeventlistener-trong-javascript-375.html#goto-h2-0)
* [2. Ví dụ hàm removeEventListener() trong Javascript](http://freetuts.net/ham-removeeventlistener-trong-javascript-375.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/ham-removeeventlistener-trong-javascript-375.html#goto-h2-2)

Nếu như hàm addEventListener() dùng để **thêm một hành động cho một sự kiện** thì hàm removeEventListener() lại có tác dụng ngược lại, nó sẽ xóa đi một hành động đã được gán cho một sự kiện.

**1. Hàm removeEventListener() trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Ví dụ bạn gán hành động validate form cho sự kiện submit của form, sau đó bạn không muốn validate nữa thì bạn sẽ có hai giải pháp. Một là bạn sẽ phải xóa đi đoạn code validate đó và hai là bạn chỉ cần sử dụng hàm removeEventListener() để xóa hành động validate.

**Cú pháp**:

|  |  |
| --- | --- |
| 1 | object.removeEventListener("click", some\_action); |

Chính vì nó sẽ xóa một hành động cho nên bắt buộc phải biết tên hành động đó là gì, vì vậy ta sẽ đặt hành động nó trong một [hàm](http://freetuts.net/ham-va-tao-ham-function-trong-javascript-274.html) để nhận diện.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | // Lấy đối tượng  var object = document.getElementById("something");    // Hanh động validate  function do\_validate()  {  // do something  }    // Thêm hành động vào sự kiện click  object.addEventListener("click", do\_validate);    // Xóa hành động validate vào sự kiện click  object.removeEventListener("click", do\_validate); |

**2. Ví dụ hàm removeEventListener() trong Javascript**

Vì hàm này khá đơn giản nên mình chỉ trình bày một ví dụ thôi nhé.

**Ví dụ**: Xây dựng một ứng dụng khi di chuyển chuột thì sẽ xuất hiện một **dãy số ngẫu nhiên**, nếu người dùng click vào button **Stop Random** thì sẽ dừng random dãy số đó.

Như vậy ta có hai thẻ HTML chính, thứ nhất là một div dùng để hiển thị dãy số ngẫu nhiên và thứ hai là một button dùng để để click **Stop Random**. Và dây là một số lưu ý trước khi làm bài:

* Đề bài yêu cầu khi **move chuột trên trang web**nên ta sẽ thêm sự kiện mousemove cho thẻ <html>.
* Dùng hàm [addEventListener()](http://freetuts.net/ham-addeventlistener-trong-javascript-374.html) để thêm sự kiện mousemove vào thẻ <html>
* Dùng đối tượng Math.random() để lấy dãy số ngẫu nhiên
* Dùng hàm removeEventListener() để xóa đi những hành động của sự kiện mousemove (xem bài [sự kiện trong javascript](http://freetuts.net/them-su-kien-event-bang-javascript-373.html)).

**Bài giải**: [XEM DEMO](http://freetuts.net/editor.html?id=306)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | <html>      <body>            <div id="result"></div>          <input type="button" value="Stop Random" id="stop\_random" />            <script language="javascript">              // Bước 1: Lấy các đối tượng              var result = document.getElementById("result");              var button = document.getElementById("stop\_random");              var html = document.getElementsByTagName("html")[0];                // Bước 2: Định nghĩa hành động hiển thị dãy số random              function do\_random()              {                  var randomString = Math.random();                  result.innerHTML = randomString;              }                // Bước 3: Thêm hành động do\_random cho sự kiện mousemove thẻ <html>,              html.addEventListener("mousemove", do\_random);                // Bước 4: Thêm sự kiện click cho button              button.addEventListener("click", function(){                  // Xóa hành động do\_random khỏi sự kiện mousemove                  html.removeEventListener("mousemove", do\_random);              });          </script>      </body>  </html> |

**3. Lời kết**

Hàm addEventListener() và removeEventListener() trong javascript sẽ không chạy ở một số trình duyệt thấp nên nếu bạn code mà không chạy thì có thể không phải do lỗi của bạn nhé.

Tóm lại hàm removeEventListener() dùng để xóa một hành động ra khỏi một sự kiện nào đó và hành động đó **phải được định nghĩa bằng một hàm**chứ không gán trực tiếp vào. Như trong ví dụ trên mình định nghĩa hành động tên là do\_random

**Bài 23: Sự kiện onload trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 20-04-2015- View: 6457

**Mục luc:**

* [1. Quá trình biên dịch code Javascript](http://freetuts.net/su-kien-onload-trong-javascript-378.html#goto-h2-0)
* [2. Sự kiện onload trong Javascript](http://freetuts.net/su-kien-onload-trong-javascript-378.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/su-kien-onload-trong-javascript-378.html#goto-h2-2)

Trong Javascript có một sự kiện được sử dụng khá nhiều đó là sự kiện onload. Nếu như bạn đã từng xem qua jQuery thì bạn cũng sẽ thấy thường trước khi viết người ta sẽ dùng cú pháp $(document).ready() thì bản chất đó chính là sự kiện onload trong Javascript.

Trước khi vào tìm hiểu về onload thì ta sẽ điểm qua một vài ý về quá trình biên dịch của Javascript đã nhé.

**1. Quá trình biên dịch code Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Cũng như các ngôn ngữ lập trình khác, Javascript sẽ chạy biên dịch từ trên xuống dưới và từ trái qua phải. Chính vì vậy khi bạn sử dụng một hàm mà **phía trên nó** không tồn tại hàm đó thì sẽ bị bái lỗi undefined ngay. Và để giải quyết vấn đề này thì ta sẽ dùng **sự kiện onload trong javascript**.

**Ví dụ**:  đoạn code này sai vì hàm do\_validate() mặc dù đã định nghĩa nhưng nó lại nằm dưới đoạn code gọi tới nó.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // Sai vì hàm do\_validate chưa được định nghĩa  var flag = do\_validate();    function do\_validate()  {      // return TRUE/FALSE;  } |

Nhưng nếu sửa lại như thế này thì sẽ đúng:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | function do\_validate()  {      // return TRUE/FALSE;  }    var flag = do\_validate(); |

Một lưu ý quan trọng nữa khi các bạn gán một hàm nào đó cho một sự kiện nào đó trong HTML thì cũng phải tuân theo quy luật trên, nghĩa là nếu bạn gán một hàm mà phía trên thẻ HTML đó không có thì nó sẽ thông báo lỗi là undefined.

**Ví dụ**: Đoạn code này cũng sai vì hàm do\_validate() ở phía trên thẻ HTML chưa được định nghĩa.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <html>    <body>      <button onclick="do\_validate()">Click me</button>      <script language="javascript">        function do\_validate()        {          // return TRUE/FALSE;        }      </script>    </body>  </html> |

Nhưng nếu bạn sửa lại như thế này thì sẽ đúng:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <html>    <body>      <script language="javascript">        function do\_validate()        {          // return TRUE/FALSE;        }      </script>      <button onclick="do\_validate()">Click me</button>    </body>  </html> |

**2. Sự kiện onload trong Javascript**

Sự kiện onload có ý nghĩa rằng khi trình duyệt đã load xong mọi thứ (image, js, css) thì những đoạn code nằm bên trong đó mới được chạy. Có một lưu ý rằng nếu bạn sử dụng onload cho một thẻ HTML nào đó thì nó sẽ có tác dụng với thẻ HTML đó thôi nhưng nếu bạn dùng cho window thì nó sẽ có tác dụng cho toàn trang.

Hay nói cách khác những đoạn code nằm bên trong sự kiện onload sẽ được **chạy sau cùng**khi mà trình JS đã được biên dịch 1 lần. Chính vì vậy nếu trong sự kiện onload bạn gọi tới một hàm nào đó thì dù bạn đặt hàm đó phía trên hay phía dưới thì đều đúng, lý do là trình biên dịch đã thực hiện dịch 1 lần rồi nên hàm đó đã tồn tại.

**Ví dụ 1**: Trong đoạn code này mình gọi hàm do\_validate() bên trong  sự kiện window.onload nên mặc dù hàm validate được đặt phía dưới nhưng vẫn đúng.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | window.onload = function()  {      do\_validate();  };    function do\_validate()  {      alert(1);  } |

**Ví dụ 2**: Nếu vẫn chưa tin thì bạn làm ví dụ sau đây, trong ví dụ này ta thực hiện alert lên thứ tự của quá trình biên dịch

[XEM DEMO](http://freetuts.net/editor.html?id=307)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | alert(1);    window.onload = function()  {      alert(3);  };    alert(2); |

Mặc dù đoạn code alert(3) nằm ở vị trí thứ hai nhưng nó lại xuất hiện cuối cùng.

**3. Lời kết**

Như vậy sự kiện onload trong Javascript được sử dụng khá nhiều bởi nó được xử lý sau cùng nên sẽ tránh được khá nhiều lỗi undefined, tuy nhiên không phải lúc nào sử dụng nó đều tốt mà bạn phải cân nhắc nhé. Và đáng lẽ ra bài này mình viết ở những bài đầu tiên nhưng mình sợ khó giải thích bởi vì nó liên quan đến [hàm trong javascript](http://freetuts.net/ham-va-tao-ham-function-trong-javascript-274.html), [sự kiện trong javascript](http://freetuts.net/su-kien-event-trong-javascript-368.html).

Cuối cùng chúc các bạn học tốt nhé!

**Bài 24: Biến toàn cục và biến cục bộ trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 21-04-2015- View: 7887

**Mục luc:**

* [1. Biến cục bộ trong Javascript](http://freetuts.net/bien-toan-cuc-va-bien-cuc-bo-trong-javascript-377.html#goto-h2-0)
* [2. Biến toàn cục trong Javascript](http://freetuts.net/bien-toan-cuc-va-bien-cuc-bo-trong-javascript-377.html#goto-h2-1)
* [3. Một số ví dụ biến toàn cục và biến cục bộ](http://freetuts.net/bien-toan-cuc-va-bien-cuc-bo-trong-javascript-377.html#goto-h2-2)
* [4. Lời kết](http://freetuts.net/bien-toan-cuc-va-bien-cuc-bo-trong-javascript-377.html#goto-h2-3)

**Biến toàn cục** và **biến cục bộ** là những khái niệm mà bất kì một ngôn ngữ lập trình nào đều có và Javascript cũng không ngoại lệ. Nhưng với javascript thì có một số lưu ý khác với các ngôn ngữ khác nếu trong ứng dụng của bạn trùng tên biến quá nhiều từ trong hàm cho đến ngoài hàm thì đôi khi dẫn đến những kết quả không mong muốn.

**1. Biến cục bộ trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Một biến được gọi là cục bộ khi bạn khai báo nó nằm bên trong một hàm cụ thể nào đó, lúc này biến đó sẽ không sử dụng được ở bên ngoài [hàm](http://freetuts.net/ham-va-tao-ham-function-trong-javascript-274.html).

**Ví dụ**: Trong ví dụ này đoạn code [alert()](http://freetuts.net/ham-alert-confirm-prompt-trong-javascript-264.html) biến comment ở ngoài function add\_comment()sẽ thông báo lỗi vì nó chưa được định nghĩa

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | function add\_comment()  {      var comment = "Nội dung comment";        // Đoạn code này đúng vì biến comment đã tồn tại      alert(comment);  }    // Đoạn code này sẽ sai vì không tồn tại biến comment  alert(comment); |

**2. Biến toàn cục trong Javascript**

Biến toàn cục là biến mà bạn khai báo bên ngoài và không nằm bên trong một hàm cụ thể nào cả.

**Ví dụ**: Trong ví dụ này không có đoạn code nào sai cả

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Biến toàn cục  var comment = "Nội dung comment bên ngoài";    // Hàm có sử dụng biến toàn cục  function add\_comment()  {      alert(comment);  }    // In biến toàn cục  alert(comment); |

**3. Một số ví dụ biến toàn cục và biến cục bộ**

Cách sử dụng biến toàn cục và biến cục bộ trong javascript có rất nhiều lưu ý nên trong phần này chúng ta sẽ tìm hiểu thông qua các ví dụ nhé.

Nếu trong hàm có sử dụng từ khóa var và để tạo một biến và tên của biến đó **đã tồn tại ở bên ngoài** (toàn cục) thì lúc này bên trong hàm nó sẽ hiểu là đang sử dụng biến cục bộ chứ không ảnh hưởng gì biến bên ngoài cả.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=308)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <html>      <body>          <script language="javascript">              // Biến toàn cục            var comment = "Nội dung comment toàn cục";              // Hàm có sử dụng biến toàn cục            function add\_comment()            {               var comment = "Nội dung comment cục bộ";               alert(comment);            }              // Gọi fuction comment            add\_comment();              // In biến toàn cục            alert(comment);            </script>      </body>  </html> |

Nếu bạn không sử dụng từ khóa var để tạo tên biến trong hàm thì nó sẽ sử dụng biến toàn cục nên mọi thay đổi của biến đó trong hàm sẽ bị ảnh hưởng ra ngoài hàm.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=309)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <html>      <body>          <script language="javascript">              // Biến toàn cục            var comment = "Nội dung comment trước khi thay đổi";              // Hàm có sử dụng biến toàn cục            function add\_comment()            {               comment = "Nội dung comment đa bị thay đổi";               alert(comment);            }              // Gọi fuction comment            add\_comment();              // In biến toàn cục            alert(comment);            </script>      </body>  </html> |

Nếu liên tưởng qua ngôn ngữ PHP thì không thể thực hiện như hai ví dụ này được mà nó phải thông qua một từ khóa nữa đó là global thì trong hàm mới sử dụng được biến toàn cục ở ngoài hàm.

**4. Lời kết**

Tóm lại biến toàn cục là biến được khai báo bên ngoài hàm  và được sử dụng ở tất cả các vị trí. Biến cục bộ là biến khai báo ở trong hàm và chỉ sử dụng được trong nội bộ của hàm đó thôi, sau khi hàm thực thi xong thì ngay lập tức nó sẽ bị xóa khỏi bộ nhớ. Và một chú ý là ở phần ví dụ bạn cần hiểu được trường hợp ghi đè tên biến thông qua từ khóa var ở trong hàm nhé.

**Bài 25: Sử dụng Firebug để kiểm tra lỗi Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 21-04-2015- View: 5369

**Mục luc:**

* [1. Cài đặt Firebug cho Firefox](http://freetuts.net/su-dung-firebug-de-kiem-tra-loi-javascript-380.html#goto-h2-0)
* [2. Sử dụng Firebug để kiểm tra lỗi Javascript](http://freetuts.net/su-dung-firebug-de-kiem-tra-loi-javascript-380.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/su-dung-firebug-de-kiem-tra-loi-javascript-380.html#goto-h2-2)

Việt phát hiện lỗi trong Javascript tương đối phức tạp bởi trong quá trình biên dịch nếu gặp một đoạn code JS bị lỗi thì ngay lập tức nó sẽ ngưng xử lý luôn, lúc này các đoạn code JS bên dưới sẽ không được thực hiện được. Chính vì vậy việc tìm lỗi trong javascript tương đối quan trọng và nó rất khó khăn với những bạn newbie mới **học javascript.** Thấy được vấn đề này nên trong bài 25 này mình sẽ trình bày một các debug lỗi trong Javascript bằng cách sử dụng **addon Firebug** của trình duyệt Firefox.

**Lưu ý**: Vì mỗi người có một sở thích và một thói quen sử dụng trình duyệt để chạy riêng nên trong bài này thói quen của mình là sử dụng Firefox, vì vậy nếu bạn đang sử dụng Chrome thì nó cũng tương tự thôi nhé.

**1. Cài đặt Firebug cho Firefox**

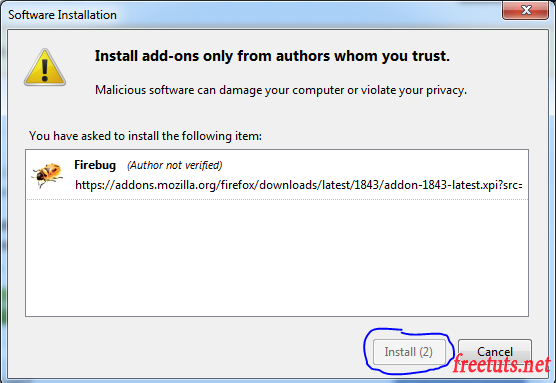
Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Để cài đặt Firebug cho Firefox bạn thực hiện các bước sau

**Bước 1**: Bạn click vào [link này](https://addons.mozilla.org/en-us/firefox/addon/firebug/),  sau đó click vào button màu xanh **Add to Firefox**.



**Bước 2**: Sau khi download về xong nó sẽ xuất hiện một popup, bạn chờ nó chạy 3 giây xong và click vào button **Install**.



**Bước 3**: Sau khi cài đặt xong đôi khi nó bắt bạn restart lại trình duyệt thì bạn hãy làm theo nhé. Sau khi thực hiện xong các bước bạn nhìn góc trên bên phải sẽ thấy biểu tượng **hình con bọ** sau:



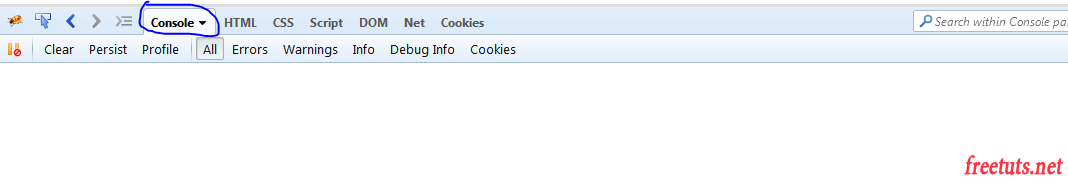
Ok như vậy là bạn đã **cài đăt Firebug cho Firefox** thành công rồi đấy. Sau này muốn sử dụng thì bạn chỉ việc click vào biển tượng con bọ hoặc nhấn **F12** để bật Firebug, lúc này  giao diện của Firebug sẽ như hình sau:



**2. Sử dụng Firebug để kiểm tra lỗi Javascript**

Bây giờ chúng ta sẽ học các **kiểm tra lỗi Javascript bằng Firebug**.

Tất cả thông tin lỗi Javascript đều nằm trong tab Console nên bạn click vào nó nhé:

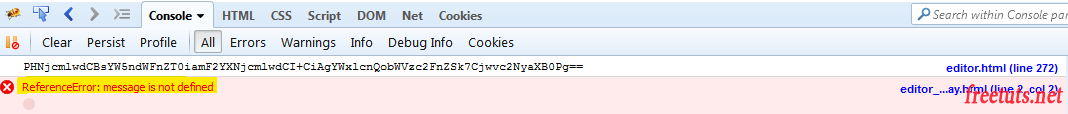


Ok, và để rõ ràng hơn thì ta sẽ học thông qua các ví dụ và tất cả ví dụ mình đều đặt trong file index.html nhé các bạn.

**Ví dụ 1**: Lỗi sử dụng biến chưa định nghĩa

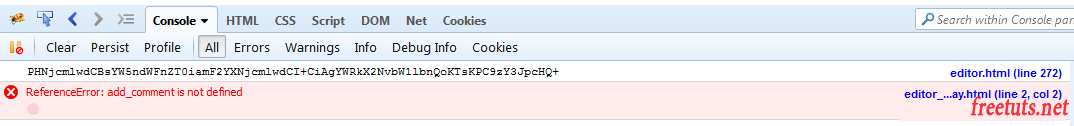
Trong ví dụ này mình đã sử dụng hàm [alert()](http://freetuts.net/ham-alert-confirm-prompt-trong-javascript-264.html) để xuất giá trị biến message nhưng bị lỗi biếnmessage chưa được định nghĩa.

|  |  |
| --- | --- |
| 1  2  3 | <script language="javascript">    alert(message);  </script> |



**Ví dụ 2**: Lỗi sử dụng hàm chưa được định nghĩa

Tương tự như **ví dụ 1**, trong ví dụ này ta đã sử dụng một hàm add\_comment() chưa được định nghĩa (xem bài [hàm trong javascript](http://freetuts.net/ham-va-tao-ham-function-trong-javascript-274.html) nếu bạn chưa biết về hàm).



Ok trên là hai ví dụ căn bản nhất mà bạn có thể học cách **debug Javascript bằng Firebug**.

**3. Lời kết**

Làm việc với Javascript mà không sử dụng Firebug giống như bạn sử dụng cuốc để cày ruộng vậy đó, nên thay vì sử dụng cuốc thì bạn hãy sử dụng công cụ khác như máy cày đất hoặc trâu cày. Nói cho vui thôi nhưng thực sự mình muốn nhấn mạnh sự lợi hại của Firebug, bạn hoàn toàn kiểm soát được các lỗi trong Javascript mà bạn không tài nào hiểu tại sao code lại không chạy như: **gõ sai tên biến, thay vì gõ comment thì bạn gõ comnent** :D.

Ok ngoài cách kiểm soát lỗi này ra thì trong bài tới mình sẽ hướng dẫn bạn sử dụng hàm console.log kết hợp với Firebug để kiểm tra dữ liệu trong javascript.

**Bài 26: Lệnh console.log kết hợp với Firebug trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 23-04-2015- View: 4349

**Mục luc:**

* [1. Lệnh console.log() trong javascript](http://freetuts.net/lenh-consolelog-ket-hop-voi-firebug-trong-javascript-381.html#goto-h2-0)
* [2. Lời kết](http://freetuts.net/lenh-consolelog-ket-hop-voi-firebug-trong-javascript-381.html#goto-h2-1)

Trong Javascript có một hàm thường được sử dụng để debug đó là hàm console.log(), nó có nhiệm vụ **show ra giá trị** của tất cả các loại dữ liệu như number, integer, array, object, chính vì vậy khi muốn biết trong một biến đó có giá trị gì thì ta sẽ sử dụng hàm console.log().

Quay lại bài trước mình đã giới thiệu với các bạn cách sử dụng [firebug để kiểm tra lỗi Javascript](http://freetuts.net/su-dung-firebug-de-kiem-tra-loi-javascript-380.html), vậy thì trong bài này chúng ta sẽ kết hợp Firebug và hàm console.log() luôn nhé.

**1. Lệnh console.log() trong javascript**

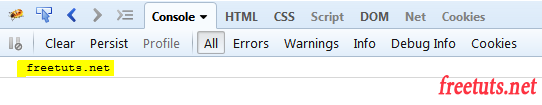
Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Lệnh console.log() có cú pháp console.log(value), trong đó value là biến hoặc giá trị mà bạn muốn in ra.

**Ví dụ 1**: console.log() một biến bình thường

|  |  |
| --- | --- |
| 1  2 | var website = 'freetuts.net';  console.log(website); |

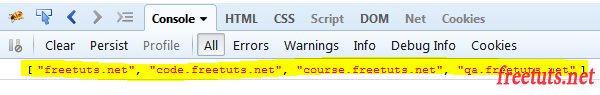
Kết quả:



**Ví dụ 2**: console.log() một mảng

|  |  |
| --- | --- |
| 1  2 | var websites = ["freetuts.net", "freetuts.net", "course.freetuts.net", "qa.freetuts.net"];  console.log(websites); |

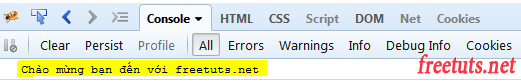
Kết quả:



**Ví dụ 3**: console.log() một giá trị

|  |  |
| --- | --- |
| 1 | console.log("Chào mừng bạn đến với freetuts.net"); |

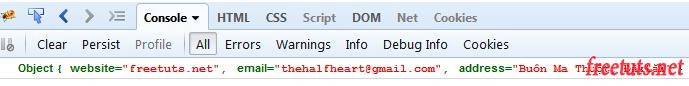
Kết quả:



**Ví dụ 4**: console.log() một object

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var info = {      website : "freetuts.net",      email : "thehalfheart@gmail.com",      address : "Buôn Ma Thuột, ĐăkLăk"  };  console.log(info); |

Kết quả:



Như vậy hàm console.log() sẽ in ra tất cả các giá trị của một biến nên bạn dễ dàng debug phải không nào. Mình thì rất hay sử dụng nó để xem kết quả [ajax](http://freetuts.net/tag/ajax-toan-tap) gửi về có những gì để từ đó đưa ra những hướng giải quyết bài toán chính xác hơn.

**2. Lời kết**

Tóm lại trong quá trình làm việc với Javscript bạn không thể bỏ qua bước debug được nên bắt buộc phải tìm các giải pháp và hàm console.log() chính là một giải pháp trong đó. Bạn cứ tưởng tượng rằng nếu bạn sử dụng hàm [alert()](http://freetuts.net/ham-alert-confirm-prompt-trong-javascript-264.html) thì giả sử website của bạn đang hoạt động và khi khách hàng vào trang web sẽ xuất hiện những popup debug không mong muốn, chính vì vậy ta sử dụng console để xem kết quả.

**Bài 27: Lệnh break - continue trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 23-04-2015- View: 3419

**Mục luc:**

* [1. Lệnh break trong Javascript](http://freetuts.net/lenh-break-continue-trong-javascript-382.html#goto-h2-0)
* [2. Lệnh continue trong Javascript](http://freetuts.net/lenh-break-continue-trong-javascript-382.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/lenh-break-continue-trong-javascript-382.html#goto-h2-2)

Tiêp tục với loạt serie học Javascript toàn tập thì trong bài này chúng ta sẽ tìm hiểu đến hai lệnh được sử dụng cũng rất thường xuyên đó là lệnh break và **continue trong javascript**.

Thực sự mà nói thì mình viết nhiều serive về các ngôn ngữ lập trình như PHP, Javascript, .. thì đều có lệnh này nên đôi khi trình bày lại cũng thấy lười, nhưng đành phải chấp nhận thôi vì phải đặt mình vào người đọc :D.

**1. Lệnh break trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Lệnh break có tác dụng dừng vòng lặp cho dù điều kiện của vòng lặp **vẫn đang đúng**, hay nói cách khác là nó thoát khỏi vòng lặp một cách đột xuất và không quan tâm đến điều kiện lặp.

Lệnh break có thể sử dụng với mọi vòng lặp như [vòng lặp for](http://freetuts.net/bai-04-vong-lap-trong-python-685.html), while và do while, each, ...

**Ví dụ 1**: Vòng lặp for lặp từ 1 tới 10 và bị dừng tại vòng lặp thứ 5

[XEM DEMO](http://freetuts.net/editor.html?id=310)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | for (var i = 1; i <= 10; i++)  {      document.write(i + " - ");      if (i == 5) {          document.write("Vòng lặp bị dừng");          break;      }  } |

**Ví dụ 2**: Vòng lặp while bị nhảy ra khỏi vòng lặp khi biến i chia hết cho 9

[XEM DEMO](http://freetuts.net/editor.html?id=311)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | var i = 1;    while (i <= 1000)  {    document.write(i + " - ");      if (i % 9 == 0) {        document.write("Vòng lặp bị dừng");        break;    }      i++;  } |

**2. Lệnh continue trong Javascript**

Khác hoàn toàn với lệnh break, lệnh continue trong javascript có tác dụng bỏ qua **một bước lặp** nào đó, nghĩa là lúc gặp lệnh continue thì tất cả những đoạn code nằm bên dưới sẽ không được thực hiện mà nó sẽ nhảy qua vòng lặp mới luôn.

**Ví dụ 1**: Vòng lặp for bỏ qua đoạn code in ra giá trị 5

[XEM DEMO](http://freetuts.net/editor.html?id=312)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | for (var i = 1; i <= 10; i++)  {    if (i == 5) {        continue;    }    document.write(i + " - ");  } |

**Ví dụ 2**: Vòng lặp while bỏ qua bước lặp nếu i chia hết cho 9

[XEM DEMO](http://freetuts.net/editor.html?id=313)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | var i = 1;    while (i <= 100)  {      if (i % 9 == 0) {          i++;          continue;      }        document.write(i + " - ");      i++;  } |

**3. Lời kết**

**Lưu ý**: Ngoài [vòng lặp for](http://freetuts.net/vong-lap-for-trong-javascript-270.html) và [vòng lặp while](http://freetuts.net/vong-lap-while-do-while-trong-javascript-271.html) ra thì hai lệnh này có thể được dùng với tất cả các vòng lặp khác

Lệnh continue và lệnh **break trong javascript** giúp cho chương trình có thể hoạt động một cách lắc léo hơn vì vậy nó cũng rất hay được sử dụng, tuy nhiên về mức độ thường xuyên thì lệnh break được sử dụng nhiều hơn lệnh continue.

**Bài 28: DOM Nodes trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 24-04-2015- View: 4710

**Mục luc:**

* [1. DOM Node - document.createElement()](http://freetuts.net/dom-nodes-trong-javascript-383.html#goto-h2-0)
* [2. DOM Node - document.createTextNode()](http://freetuts.net/dom-nodes-trong-javascript-383.html#goto-h2-1)
* [3. DOM Node - các phương thức khác](http://freetuts.net/dom-nodes-trong-javascript-383.html#goto-h2-2)
  + [Phương thức appendChild()](http://freetuts.net/dom-nodes-trong-javascript-383.html#goto-h3-0)
  + [Phương thức insertBefore()](http://freetuts.net/dom-nodes-trong-javascript-383.html#goto-h3-1)
  + [Phương thức removeChild()](http://freetuts.net/dom-nodes-trong-javascript-383.html#goto-h3-2)
  + [Phương thức replaceChild()](http://freetuts.net/dom-nodes-trong-javascript-383.html#goto-h3-3)
* [4. Lời kết](http://freetuts.net/dom-nodes-trong-javascript-383.html#goto-h2-3)

Trong serie này chúng ta đã học về [DOM](http://freetuts.net/dom-la-gi-cac-loai-dom-trong-javascript-366.html) khá nhiều rồi như DOM Element, DOM HTML, DOM CSS nhưng có lẽ bạn vẫn chưa hiểu bản chất của các đối tượng trả về khi sử dụng DOM Element để truy vấn đó chính là DOM Nodes.

Vậy trong bài này chúng ta sẽ tìm hiểu khái niệm **DOM Nodes là gì** thông qua các ví dụ và bạn sẽ học luôn một số phương thức thường được sử dụng trong DOM Nodes.

**1. DOM Node - document.createElement()**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Khi bạn sử dụng [DOM Element](http://freetuts.net/dom-element-trong-javascript-367.html) để truy vấn tới một đối tượng HTML nào đó thì kết quả nó sẽ trả về một object và object đó ta gọi là DOM Nodes.

**Ví dụ**:

|  |  |
| --- | --- |
| 1 | var node = document.getElementById("some-id"); |

Với cách này bắt buộc phải tồn tại một đối tượng HTML đang hiển thị trên website thì mới khởi tạo thành công. Giả sử nếu bạn muốn tạo một Node mới hoàn toàn v không liên quan tới những thẻ HTML đang hiển thị trên website thì làm thế nào? Rất đơn giản chúng ta sẽ sử dụng phương thức document.createElement() với tham số truyền vào là tên của thẻ HTML cần tạo.

|  |  |
| --- | --- |
| 1 | var p = document.createElement("p"); |

Sau khi khởi tạo xong bạn hoàn toàn có thể sử dụng các phương thức, thuộc tính của [DOM HTML](http://freetuts.net/dom-html-trong-javascript-370.html), [DOM CSS](http://freetuts.net/dom-css-trong-javascript-371.html).

|  |  |
| --- | --- |
| 1 | p.innerHTML = "Học DOM Nodes tại freetuts.net" |

Để thêm Node này vào trang web thì chúng ta sử dụng phương thức appendChild (sẽ học ở bên dưới). Giả sử tôi thêm vào thẻ body thì làm như sau:

|  |  |
| --- | --- |
| 1 | document.getElementsByTagName('body')[0].appendChild(p); |

**Và đây là toàn bộ code cho ví dụ này**:

[XEM DEMO](http://freetuts.net/editor.html?id=314)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <html>      <body>          <script language="javascript">              // Tạo mới một thẻ p            var p = document.createElement("p");              // Thêm nội dung HTML vào thẻ p            p.innerHTML = "Học DOM Nodes tại freetuts.net"              // Đưa thẻ P vào trong thẻ body            document.getElementsByTagName('body')[0].appendChild(p);            </script>      </body>  </html> |

Ok bây giờ chúng ta qua các phần tiếp theo để tìm hiểu thêm một số phương thức hay sử dụng trong DOM Nodes nhé.

**2. DOM Node - document.createTextNode()**

Text node là một node đặc biệt, nó không phải là một thẻ HTML thông thường mà chỉ là một chuỗi (string) nên thông thường chúng ta sử dụng nó để thay thế cách gán thông thường node.innerHTML.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=319)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <html>      <body>          <script language="javascript">              // Tạo mới một thẻ p            var p = document.createElement("p");              // Tạo text node            var text = document.createTextNode("Học DOM Nodes tại freetuts.net");              // Thêm nội dung HTML vào thẻ p            p.appendChild(text);              // Đưa thẻ P vào trong thẻ body            document.getElementsByTagName('body')[0].appendChild(p);            </script>      </body>  </html> |

Trong ví dụ này thay vì sử dụng cách thông thường như ví dụ ở phần 1 thì mình đã thay thế bằng cách sử dụng text node.

**3. DOM Node - các phương thức khác**

**Phương thức appendChild()**

Dùng để thêm (**bổ sung**) vào vị trí cuối cùng của đối tượng một thẻ HTML nào đó.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=315)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <html>      <body>          <div id="TOP">              Xin chào các bạn!          </div>            <input type="button" value="Append" id="btn-append"/>            <script language="javascript">            // Lấy button            var button = document.getElementById("btn-append");              // Thêm sự kiện click cho button            button.addEventListener("click", function(){              // Tạo mới một thẻ h1              var h1 = document.createElement("h1");                // Thêm nội dung HTML vào thẻ h1              h1.innerHTML = "Học DOM Nodes tại freetuts.net"                // Đưa thẻ h1 vào trong thẻ div có id=TOP              document.getElementById('TOP').appendChild(h1);            });          </script>      </body>  </html> |

**Phương thức insertBefore()**

Được dùng để thêm một Node vào đằng trước một node con nào đó. Phương thức này có hai tham số truyền vào insertBefore(node\_insert, node\_child), trong đó:

* node\_insert là node bạn muốn thêm vào
* node\_child là node con mà bạn muốn thêm vào đằng trước nó.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=316)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | <html>      <body>          <div id="content">              <h5>Xin chào các bạn</h5>              <h5>chúc vui vẻ khi học bài</h5>          </div>            <input type="button" value="View" id="btn-view"/>            <script language="javascript">            // Lấy button            var button = document.getElementById("btn-view");              // Thêm sự kiện click cho button            button.addEventListener("click", function(){              // Tạo mới một thẻ h1              var h1 = document.createElement("h1");                // Thêm nội dung HTML vào thẻ h1              h1.innerHTML = "Chào mừng bạn đến với freetuts.net"                // Lấy thẻ ngoài cùng              var element = document.getElementById("content");                // Lấy thẻ cần isertBefore              var element\_child = document.getElementsByTagName("h5")[1];                // Insert Before              element.insertBefore(h1, element\_child);            });          </script>      </body>  </html> |

**Phương thức removeChild()**

Được dùng để xóa một node con ra khỏi node hiện tại.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=317)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <html>      <body>          <div id="content">              <h5>chúc vui vẻ khi học bài</h5>          </div>            <input type="button" value="Remove" id="btn-remove"/>            <script language="javascript">            // Lấy button            var button = document.getElementById("btn-remove");              // Thêm sự kiện click cho button            button.addEventListener("click", function(){                // Lấy thẻ cần remove              var need\_remove = document.getElementsByTagName("h5")[0];                // Remove              document.getElementById("content").removeChild(need\_remove);            });          </script>      </body>  </html> |

**Phương thức replaceChild()**

Dùng để replace (thay thế) một node con nào đó bằng một node khác mới hoàn toàn.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=318)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | <html>      <body>          <div id="content">              <h5>chúc vui vẻ khi học bài</h5>          </div>            <input type="button" value="Replace" id="btn-replace"/>            <script language="javascript">            // Lấy button            var button = document.getElementById("btn-replace");              // Thêm sự kiện click cho button            button.addEventListener("click", function(){                // Tạo mới một thẻ              var p = document.createElement("h1");              p.innerHTML = "Xin chào!";                // Lấy thẻ cần replace              var replace = document.getElementsByTagName("h5")[0];                // Replace              document.getElementById("content").replaceChild(p, replace);            });          </script>      </body>  </html> |

**4. Lời kết**

Việc sử dụng thành thạo các phương thức xử lý **DOM Node** rất quan trọng khi bạn làm việc với các thẻ HTML bằng Javascript. Nếu bạn sử dụng một JS Library như jQuery thì điều này khá đơn giản bởi vì nó đã viết ra những hàm cho chúng ta sử dụng sẵn nên hầu hết các lập trình viên đều không quan tâm đến những kiến thức thuần javascript như thế này.

Ngoài các phương thức mình liệt kê trên thì còn khá nhiều phương thức nên bạn có thể sử dụng Google để tìm kiếm trong quá trình học và làm việc của mình. Bản thân mình cũng vậy thôi không tài nào nhớ hết được nhung phải biết từ khóa để tìm kiếm.

**Bài 29: BOM là gì? BOM trong javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 24-04-2015- View: 4335

**Mục luc:**

* [1. BOM là gì?](http://freetuts.net/bom-la-gi-bom-trong-javascript-384.html#goto-h2-0)
* [2. Lời kết](http://freetuts.net/bom-la-gi-bom-trong-javascript-384.html#goto-h2-1)

**1. BOM là gì?**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

BOM là chữ viết tắt của **B**rowser **O**bject **M**odel, hay còn gọi là các đối tượng liên quan đến trình duyệt browser. Mỗi browser sẽ có những đối tượng khác nhau nên nó không có một chuẩn chung nào cả, tuy nhiên để có tính thống nhất giữa các trình duyệt thì người ta quy ước ra các loại BOM sau:

* [window](http://freetuts.net/bom-window-trong-javascript-385.html)
* screen
* [location](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html)
* [history](http://freetuts.net/bom-history-trong-javascript-387.htm)
* navigator
* popup
* timing
* cookies

Trong các đối tượng DOM trên có phân cấp lẫn nhau và trong đó window là cấp cao nhất vì nó đại diện cho browser. Ví dụ khi bạn muốn truy cập tới document thì bạn sẽ viêt là window.document, tuy nhiên vẫn có cách ghi tắt là document, muốn truy cập tới cookie thì ta viết window.document.cookie (viết tắt document.cookie), ...

Trong danh sách trên thì popup chúng ta đã được học ở bài [alert() - confirm() - prompt()](http://freetuts.net/ham-alert-confirm-prompt-trong-javascript-264.html) rồi nên mình sẽ không trình bày lại. Còn 7 các đối tượng còn lại sẽ được trình bày ở 7 bài tiếp theo.

**2. Lời kết**

Cho dù bạn sử dụng một thư viện JS khác như jQuery thì bạn cũng không thể bỏ qua các đối tượng BOM được nên việc nắm bắt các đối tượng BOM này là điều bắt buộc khi làm việc với Javascript. Vậy bạn không nên bỏ qua loạt bài này hoặc nếu bận quá thì  đọc lướt qua để biết là trong JS có chức năng này và cho dù bạn quên thì bạn vẫn biết tìm lại kiến thức ở đâu.

Bài này mình chỉ giới thiệu **BOM là gì** và liệt kê danh sách các đối tượng BOM trong Javascript nên sẽ dừng tại đây, chúc các bạn học Javascript vui vẻ và với niềm đam mê đó bạn sẽ thành công.

**Bài 30: BOM - Window trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 25-04-2015- View: 5327

**Mục luc:**

* [1. Xác định kích thước của trình duyệt](http://freetuts.net/bom-window-trong-javascript-385.html#goto-h2-0)
* [2. Thao tác với một cửa sổ window](http://freetuts.net/bom-window-trong-javascript-385.html#goto-h2-1)
  + [Mở một cưa sổ với lệnh window.open()](http://freetuts.net/bom-window-trong-javascript-385.html#goto-h3-0)
  + [Đóng một cửa số với lệnh window.close()](http://freetuts.net/bom-window-trong-javascript-385.html#goto-h3-1)
  + [Di chuyển cửa sổ với lệnh window.moveTo()](http://freetuts.net/bom-window-trong-javascript-385.html#goto-h3-2)
  + [Resize cửa sổ với lệnh window.resizeTo()](http://freetuts.net/bom-window-trong-javascript-385.html#goto-h3-3)
* [3. Lời kết](http://freetuts.net/bom-window-trong-javascript-385.html#goto-h2-2)

Đây là bài đầu tiên trong loạt bài giới thiệu về [DOM trong Javascript](http://freetuts.net/bom-la-gi-bom-trong-javascript-384.html) và trong bài này chúng ta sẽ tìm hiểu đến đối tượng Window. Đối tượng Window được hỗ trợ hầu hết các trình duyệt và mỗi tab trình duyệt sẽ có một đối tượng Window riêng và không liên quan tới nhau.

Đối tượng Window có rất nhiều thuộc tính và phương thức riêng. Ví dụ đối tượng documentlà một thuộc tính của window (window.document), vì vậy nên ta có thể ví Window là một đối tượng toàn cục và ở đâu trong website cũng có thể sử dụng được.

Bây giờ chúng ta sẽ tìm hiểu các thuộc tính và phương thức của đối tượng BOM Window nhé.

**1. Xác định kích thước của trình duyệt**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Để lấy kích thước chiều cao và chiều rộng của trình duyệt thì chúng ta sử dụng đối tượng window, tuy nhiên với mỗi trình duyệt thì có những cách lấy khác nhau.

**Đối với Internet Explorer, Chrome, Firefox, Opera, và Safari thì cú pháp như sau**:

|  |  |
| --- | --- |
| 1  2  3  4  5 | // lấy chiều cao  var heightBrowser = window.innerHeight;    // lấy chiều rộng  var widthBrowser = window.innerWidth; |

**Đối với Internet Explorer các version 5,6,7,8 thì cú pháp như sau**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // Lấy chiều cao  var height = document.documentElement.clientHeight;  // hoặc  var height = document.body.clientHeight;    // Lấy chiều rộng  var width = document.documentElement.clientWidth;  // hoặc  var width = document.body.clientWidth; |

Việc tương thích với các trình duyệt quá rắc rối phải không nào? Vậy có cách nào giải quyết vấn đề này không? Câu trả lời là có bằng cách sử dụng một mẹo nhỏ như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var width = window.innerWidth          || document.documentElement.clientWidth          || document.body.clientWidth;    var height = window.innerHeight          || document.documentElement.clientHeight          || document.body.clientHeight; |

Hai dấu || có ý nghĩa rằng nếu vế trái không thực hiện được thì nó sẽ lấy vế phải và cứ như vậy cho tới khi lấy được thì dừng.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=320)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var width = window.innerWidth  || document.documentElement.clientWidth  || document.body.clientWidth;    var height = window.innerHeight  || document.documentElement.clientHeight  || document.body.clientHeight;    document.write("Trình duyệt có chiều rộng là: " + width + "<br/>");  document.write("Trình duyệt có chiều cao là: " + height + "<br/>"); |

**2. Thao tác với một cửa sổ window**

Đối tượng window cung cấp một số phương thức giúp thao tác với trình duyệt dễ dàng hơn.

**Mở một cưa sổ với lệnh window.open()**

**Cú pháp**: window.open(url, name, options)

**Trong đó**:

* **url**: là đường dẫn website bạn muốn mở
* **name**: là tên bạn đặt cho cửa sổ này
* **options**: là một chuỗi các thông số được cách nhau bởi dấu phẩy, sau đây là các thông số thông dụng:
  + **height=pixels** : chiều cao của cửa sổ
  + **width=pixels**: chiều rộng của cửa sổ
  + **top=pixels**: vị trí hiển thị cửa sổ so với lề trên
  + **left=pixels**: vị trí hiển thị cửa sổ so với lề trái
  + **menubar=yes|no|1|0**: có hiển thị thanh menu hay không?
  + **resizable=yes|no|1|0**: có hiển thị biểu tượng resize cửa sổ hay không?
  + **scrollbars=yes|no|1|0**: có hiển thị thanh cuộn hay không?
  + **status=yes|no|1|0**: có hiển thị thanh trạng thái hay không?
  + **titlebar=yes|no|1|0**: có hiển thị titlebar hay không?
  + **toolbar=yes|no|1|0**: có hiển thị toolbar hay không?
  + **fullscreen=yes|no|1|0**: có hiển thị biểu tượng fullscreen hay không?

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=321)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <html>      <body>        <script language="javascript">            var windowChild = null;            function openWindow()          {            windowChild = window.open('[http://freetuts.net](http://freetuts.net/)', "windowChild", "width=500, height=500");              return false;          }          </script>        <a href="#" onclick="return openWindow()">Open</a>      </body>  </html>  </script> |

**Đóng một cửa số với lệnh window.close()**

Sau khi mở cửa sổ thì để đóng cửa sổ đó chúng ta sử dụng lệnh windowObj.close(), trong đó windowObj là cửa sổ mà ta sử dụng lệnh window.open() tạo ra.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=322)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <html>      <body>        <script language="javascript">            var windowChild = null;            function openWindow()          {              windowChild = window.open('[http://freetuts.net](http://freetuts.net/)', "windowChild", "width=500, height=500");              return false;          }            function closeWindow()          {              windowChild.close()              return false;          }        </script>        <a href="#" onclick="return openWindow()">Open</a>        <a href="#" onclick="return closeWindow()">Close</a>      </body>  </html> |

**Di chuyển cửa sổ với lệnh window.moveTo()**

Sau khi mở một cửa sổ nếu muốn di chuyển nó thì ta dùng lệnh windowObj.moveTo(top, left), trong đó:

* **top**: là số pixels so với lề trên
* **left**: là số pixels so với lề trái

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <html>      <body>        <script language="javascript">            var windowChild = null;            function openWindow()          {              windowChild = window.open('[http://freetuts.net](http://freetuts.net/)', "windowChild", "width=500, height=500");              return false;          }            function moveWindow()          {              windowChild.moveTo(500, 100);              windowChild.focus();              return false;          }        </script>        <a href="#" onclick="return openWindow()">Open</a>        <a href="#" onclick="return moveWindow()">Move</a>      </body>  </html> |

**Resize cửa sổ với lệnh window.resizeTo()**

Lúc mở cửa sổ bạn sẽ thiết lập height và with cho window, tuy nhiên nếu sau khi mở bạn muốn thay đổi thì sử dụng hàm windowObj.resizeTo(width, height), trong đó:

* **width**: chiều rộng của cửa sổ
* **height**: chiều cao của cửa sổ

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <html>      <body>        <script language="javascript">            var windowChild = null;            function openWindow()          {              windowChild = window.open('[http://freetuts.net](http://freetuts.net/)', "windowChild", "width=500, height=500");              return false;          }            function resizeWindow()          {              windowChild.resizeTo(1000, 1000);              windowChild.focus();              return false;          }        </script>        <a href="#" onclick="return openWindow()">Open</a>        <a href="#" onclick="return resizeWindow()">Resize</a>      </body>  </html> |

**3. Lời kết**

Trong các ví dụ ở phần moveTo() và resizeTo() mình không có demo bởi vì trình duyệt đã chặn hai phương thức này nên bạn hãy copy code vào file của bạn rồi chạy nhé.

Vẫn còn một số phương thức nữa nhưng bài này mình chỉ đề cập tới bấy nhiêu là quá đủ cho bạn học rồi, nếu cần thiết thì bạn có thể sử dụng google để tra cứu thêm. Và việc sử dụng đối tượng window để mở các cửa sổ không còn thông dụng nữa mà thay vào đó người ta sử dụng một kỹ thuật khác đó là xây dựng Popup nhìn chuyên nghiệp và đẹp hơn.

**Bài 31: BOM - Location điều hướng và xử lý URL trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 25-04-2015- View: 7984

**Mục luc:**

* [1. Location trong Javascript](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html#goto-h2-0)
* [2. Các phương thức của Location](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html#goto-h2-1)
  + [window.location.reload(url) - tải lại trang web](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html#goto-h3-0)
  + [window.location.replace(urr) - ghi đè trang web](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html#goto-h3-1)
  + [window.location.assign(url) - load một trang mới](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html#goto-h3-2)
* [3. Các thuộc tính của Location](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html#goto-h2-2)
* [4. Lời kết](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html#goto-h2-3)

Tiếp tục serie tìm hiểu [BOM](http://freetuts.net/bom-la-gi-bom-trong-javascript-384.html) trong Javascript thì trong bài này chúng ta sẽ tìm hiểu đối tượng**location trong javascript**, đây là đối tương chuyên dùng xử lý URL của trang web và và thực hiện các thao tác liên quan tới URL như reload, redirect, ...

**1. Location trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Location là một thuộc tính của đối tượng [window](http://freetuts.net/bom-window-trong-javascript-385.html) nên khi sử dụng nó bạn phải thông qua đối tượng window và đối tượng window luôn tồn tại trong các trình duyệt hiện nay nên bạn hoàn toàn yên tâm khi sử dụng mà không cần phải khai báo.

|  |  |
| --- | --- |
| 1 | window.location |

Bây giờ chúng ta tìm hiểu các phương thức và thuộc tính của location nhé.

**2. Các phương thức của Location**

Có ba phương thức chính nằm trong đối tượng location đó là reload(), replace() và assign().

**window.location.reload(url) - tải lại trang web**

Thông thường để tại lại trang web bạn sẽ nhấn phím **F5**hoặc là clich chuột phải và chọn**Refresh page**, tuy nhiên nếu bạn **refresh bằng Javascript** thì vẫn được bằng cách sử dụng phương thức reload().

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=323)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <html>      <body>        Chào mừng bạn đến với freetuts.net        <button onclick="refreshPage()">Refresh</button>          <script language="javascript">          function refreshPage()          {              window.location.reload();          }        </script>      </body>  </html> |

**window.location.replace(urr) - ghi đè trang web**

Phương thức này ít khi sử dụng mà thay vào đó họ sử dụng cú pháp window.location.href="url". Tuy nhiên hai cách này vẫn có sự khác biệt:

* Đối với replace() thì trình duyệt sẽ KHÔNG đưa vào lịch sử.
* Đối với location.href thì trình duyệt sẽ CÓ đưa vào lịch sử

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=324)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <html>      <body>        Click vào để chuyển hướng đến freetuts.net        <button onclick="replacePage()">replace()</button>        <button onclick="hrefPage()">location.href</button>          <script language="javascript">          function replacePage()          {            window.location.replace('[http://freetuts.net](http://freetuts.net/)');          }            function hrefPage()          {            window.location.href = '[http://freetuts.net](http://freetuts.net/)';          }        </script>      </body>  </html> |

**window.location.assign(url) - load một trang mới**

Về công dụng vẫn không có gì khác với hai cách trên, tuy nhiên cách này nó có đặc điểm giống với location.href.

**Vi dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=325)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <html>      <body>        Click vào để chuyển hướng đến freetuts.net        <button onclick="assignPage()">replace()</button>          <script language="javascript">          function assignPage()          {            window.location.assign('[http://freetuts.net](http://freetuts.net/)');          }        </script>      </body>  </html> |

**3. Các thuộc tính của Location**

Ngoài các phương thức trên thì bạn có thể sử dụng Location để xử lý các thành phần liên quan đến URL như lấy phần hash, lấy phần search.

Và đây là danh sách các thuộc tính đầy đủ cho đối tượng location này:

* **hash**: thiết lập hoặc lấy phần sau dấu # của URL
* **host**: thiết lập hoặc lấy hostname và port number của URL
* **hostname**: thiết lập hoặc lấy hostname
* **href**: thiết lập hoặc lấy URL
* **origin**: trả về protocal, hostname và port number của URL
* **pathname**: thiết lập hoặc lấy path name của URL
* **port**: thiết lập hoặc lấy port của URL
* **search**: lấy phần query string của URL

**Lưu ý**: các thuộc tính trên bạn có thể dùng đẻ lấy (get) hoặc thiết lập (get).

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=326)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <html>      <body>        <script language="javascript">          document.write("hash:" +window.location.hash + "<br/>");          document.write("host:" +window.location.host + "<br/>");          document.write("hostname:" +window.location.hostname + "<br/>");          document.write("href:" +window.location.href + "<br/>");          document.write("origin:" +window.location.origin + "<br/>");          document.write("pathname:" +window.location.pathname + "<br/>");          document.write("port:" +window.location.port + "<br/>");          document.write("search:" +window.location.search + "<br/>");        </script>      </body>  </html> |

**4. Lời kết**

Các thuộc tính trên chúng ta chỉ hay sử dụng phần hash, search và href, các phần còn lại rất ít khi sử dụng. Riêng đối với hash là thành phần nằm sau dấu # nên các Framework Javascript như AngularJS hoặc các ứng dụng Ajax sử dụng để nhận diện được action hiện tại là gì.

Sắp tới ngày lễ 30/4 rồi nên có lẽ tuts mình ra sẽ không được đều nữa nên các bạn thông cảm nha, ngày đó mình sẽ về quê cùng gia đình.

**Bài 32: BOM - History trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 25-04-2015- View: 2634

**Mục luc:**

* [1. History trong Javascript](http://freetuts.net/bom-history-trong-javascript-387.html#goto-h2-0)
  + [Đếm tổng số trang đã lưu trong history](http://freetuts.net/bom-history-trong-javascript-387.html#goto-h3-0)
  + [Đi tới một trang nào đó trong history](http://freetuts.net/bom-history-trong-javascript-387.html#goto-h3-1)
* [2. Lời kết](http://freetuts.net/bom-history-trong-javascript-387.html#goto-h2-1)

Đầy là bài thứ 3 trong danh sách các bài tìm hiểu về [BOM](http://freetuts.net/bom-la-gi-bom-trong-javascript-384.html) và trong bài này chúng ta tìm hiểu đến đối tượng History trong Javascript, chúng ta chỉ tìm hiểu một số phương thức và thuộc tính hay sử dụng nhất chứ không thể trình bày một cách đầy đủ được.

**1. History trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Khi bạn lướt web thì trình duyệt sẽ lưu lại lịch sử lướt web của bạn và dựa vào lịch sử đó chúng ta có thể xem lại những ngày trước đã xem gì, và vấn đề này trình duyệt đã hỗ trợ cho chúng ta rồi (*back, forward, clear history*). Nhưng nếu bạn có nhu cầu cần sử dụng Javascript để xử lý thì vẫn có một số phương thức và thuộc tính nằm trong đối tượng History có thể giúp giải quyết vấn đề này.

Cũng như [location](http://freetuts.net/bom-location-dieu-huong-va-xu-ly-url-trong-javascript-386.html), history cũng là một thành phần con của [window](http://freetuts.net/bom-window-trong-javascript-385.html) nên chúng ta phải thông qua đối tượng window để thao tác với history.

|  |  |
| --- | --- |
| 1 | window.history |

Và sau đây chúng ta tìm hiểu một số thành phần nằm trong history nhé.

**Đếm tổng số trang đã lưu trong history**

Nếu bạn muốn đếm tổng số trang đã lưu trong lịch sử lướt web thì bạn có thể sử dụng cú pháp sau:

|  |  |
| --- | --- |
| 1 | var totalPage = window.history.length; |

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=327)

|  |  |
| --- | --- |
| 1 | alert("Tổng số trang đã lưu trong history là: " + window.history.length); |

**Đi tới một trang nào đó trong history**

Chúng ta có ba phương thức thường dùng để đến một trang trong history:

* history.back() : trở lại trang trước
* history.forward() : đi tới trang kế tiếp
* history.go(number): đi tới một trang:
  + nếu number âm thì tính từ trang hiện tại trừ đi number
  + nếu number dương thì tính từ trang hiện tại cộng với number

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // trở lại trang trước  window.history.back();    // đi tới trang kế tiếp sau khi sử dụng back()  window.history.forward()    // đi tới trang cách đây 5 lần lướt web  window.history.go(-5); |

Ok vậy là chúng ta đã tìm hiểu một số hàm quan trọng trọng history rồi đấy.

**2. Lời kết**

Hai năm nay khi xây dựng web thì mình chưa bao giờ sử dụng đối tượng history này nên có thể liệt kê nó vào danh sách những đối tượng ít sử dụng trong javascript. Tuy nhiên nếu ứng dụng của bạn có các chức năng back, forward thì nên sử dụng đối tượng này.

**Bài 33: BOM - Cookie trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 08-05-2015- View: 3634

**Mục luc:**

* [1. Cookie là gì?](http://freetuts.net/bom-cookie-trong-javascript-388.html#goto-h2-0)
* [2. Các thao tác với Cookie trong Javascript](http://freetuts.net/bom-cookie-trong-javascript-388.html#goto-h2-1)
  + [Tạo Cookie](http://freetuts.net/bom-cookie-trong-javascript-388.html#goto-h3-0)
  + [Lấy giá trị Cookie](http://freetuts.net/bom-cookie-trong-javascript-388.html#goto-h3-1)
  + [Đổi giá trị cho Cookie](http://freetuts.net/bom-cookie-trong-javascript-388.html#goto-h3-2)
  + [Xóa Cookie](http://freetuts.net/bom-cookie-trong-javascript-388.html#goto-h3-3)
* [3. Viết hàm xử lý Cookie trong Javascript](http://freetuts.net/bom-cookie-trong-javascript-388.html#goto-h2-2)
* [4. Lời kết](http://freetuts.net/bom-cookie-trong-javascript-388.html#goto-h2-3)

Cookie là một đối tượng không thể thiếu trong các trình duyệt vì hầu hết các website đều có sử dụng cookie để lưu trữ dữ liệu. Việc thao tác với cookie chúng ta phải dựa vào Javascript nên trong bài này chúng ta sẽ tìm hiểu tới **Cookie là gì** và cách sử dụng **cookie trong Javascript** như thế nào để ứng dụng cho các bài sau.

**1. Cookie là gì?**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Cookie là dữ liệu được lưu trữ trong một file text và nằm trong máy tính nên việc lưu trữ Cookie sẽ là vĩnh viễn hoặc là một thời gian cụ thể do bạn thiết lập. Tuy nhiên mỗi domain sẽ có một dung lượng Cookie tối đa có thể lưu trữ được nên hãy lưu ý trường hợp dữ liệu quá lớn.

Để hiểu rõ hơn **cookie là gì** và **tại sao lại sử dụng Cookie** thì mình đặt một trường hợp thế này: Giả sử bạn muốn lưu trữ một thông tin nào đó tại mỗi client (Browser) và các lần truy cập tiếp theo (*ngày mai, ngày một, một ngày nào đó, ..*) có thể sử dụng được thì phải làm thế nào? nếu sử dụng [Session trong PHP](http://freetuts.net/session-va-cookie-trong-php-25.html) thì chỉ lưu trữ được một khoảng thời gian nhất định. Vậy giải pháp là ta sẽ sử dụng Cookie để lưu trữ thông tin đó.

Cookie được lưu trữ ở dạng name=value, ví dụ:

website=freetuts.net

Khi trình duyệt gửi thông tin lên Server thì cookies sẽ được load và gửi kèm theo trong request, chính vì vậy trong ngôn ngữ server (PHP, JSP, ..) cũng sẽ nhận được thông tin đó. Đây cũng là thông tin để trả lời cho câu hỏi tạo sao Cookie được lưu trữ ở Client mà trên Server vẫn đọc được, và ta cũng hay sử dụng Cookie để xây dựng chức năng Remember Me khi Login.

**2. Các thao tác với Cookie trong Javascript**

Javascript có thể đọc, thêm và xóa Cookie thông qua đối tượng [BOM](http://freetuts.net/bom-la-gi-bom-trong-javascript-384.html) document.cookie.

**Tạo Cookie**

Ở phần **Cookie là gì** bạn đã biết Cookie được lưu trữ ở dạng name=value nên để tạo Cookie thì ta sử dụng chuỗi dữ liệu giống như vậy.

**Ví dụ**:

|  |  |
| --- | --- |
| 1 | document.cookie = 'website=freetuts.net'; |

Trường hợp bạn muốn lưu trữ nhiều Cookie thì chỉ việc tạo hai lần.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2 | document.cookie = 'website=freetuts.net';  document.cookie = 'email=thehalfheart@gmail.com'; |

Để thiết lập thời gian sống cho Cookie thì ta sử dụng từ khóa expires cấu trúc của chuỗi này là thời gian ở dạng UTC.

**Ví dụ**:

|  |  |
| --- | --- |
| 1 | document.cookie="website=freetuts.net; expires=Thu, 0 Dec 2015 12:00:00 UTC"; |

Theo mặc định thì khi bạn ở trang nào thì Cookie sẽ lưu với đường dẫn trang đó. Tuy nhiên nếu bạn muốn thay đổi đường dẫn thì sử dụng từ khóa path.

**Ví dụ**: Path là trang chủ

|  |  |
| --- | --- |
| 1 | document.cookie="website=freetuts.net; expires=Thu, 0 Dec 2015 12:00:00 UTC;path=\"; |

**Lấy giá trị Cookie**

Để lấy danh sách các Cookie thì ta sử dụng cú pháp như sau:

|  |  |
| --- | --- |
| 1 | var giatri = document.cookie; |

Lúc này kết quả sẽ trả về một chuỗi Cookie có cấu trúc name=value, trường hợp bạn lưu nhiều Cookie thì nó sẽ có dạng name1=value1;name2=value2.

**Đổi giá trị cho Cookie**

Để thay đổi giá trị cho Cookie thì bạn chỉ việc gán lại giá trị cho cookie.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Giá trị cũ  document.cookie = "domain=freetuts.net";    // Đổi sang giá trị mới  document.cookie = "domain=freetuts.net"; |

**Xóa Cookie**

Trong Javascript không có hàm xóa Cookie mà bạn phải sử dụng từ khóa expires để thiết lập thời gian sống cho Cookie là khoảng thời gian đã qua. Ví dụ hôm nay là ngày 8/11/2015 thì bạn chỉ việc gán expires là 7/8/2015 là đã xóa được Cookie đó.

**3. Viết hàm xử lý Cookie trong Javascript**

Vì Cookie được lưu trữ ở dạng chuỗi các key=value và các chuỗi giá trị cách nhau bởi dấu chấm phẩy ; nên để lấy một giá nào đó bạn phải xử lý chuỗi rất phức tạp, vì vậy thông thường chúng ta viết hàm tạo và lấy Cookie để sử dụng được nhiều lần.

Sau đây là hai hàm mình lấy ở W3C các bạn tham khảo.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | // Hàm thiết lập Cookie  function setCookie(cname, cvalue, exdays) {      var d = new Date();      d.setTime(d.getTime() + (exdays\*24\*60\*60\*1000));      var expires = "expires="+d.toUTCString();      document.cookie = cname + "=" + cvalue + "; " + expires;  }    // Hàm lấy Cookie  function getCookie(cname) {      var name = cname + "=";      var ca = document.cookie.split(';');      for(var i=0; i<ca.length; i++) {          var c = ca[i];          while (c.charAt(0)==' ') c = c.substring(1);          if (c.indexOf(name) == 0) return c.substring(name.length, c.length);      }      return "";  } |

Bạn chỉ việc truyền các tham số vào hàm là đã tạo và lấy được Cookie.

**4. Lời kết**

Việc sử dụng **Cookie trong Javascript** rất thường xuyên, tuy nhiên để gán và lấy giá trị Cookie thì bạn phải thông qua các giai đoạn tách chuỗi và gán chuỗi rất phức tạp. Vì vậy để tiện lợi bạn hãy viết ra những hàm riêng như phần 3 mình đã trình bày.

Ngoài ra bạn có thể sử dụng các thư viện Cookie ở bên ngoài bằng cách sử dụng Google để tìm kiếm.

**Bài 34: BOM - Window Navigator trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 08-05-2015- View: 2354

**Mục luc:**

* [1. Kiểm tra Cookie có được bật không?](http://freetuts.net/bom-window-navigator-trong-javascript-389.html#goto-h2-0)
* [2. Kiểm tra tên trình duyệt đang sử dụng](http://freetuts.net/bom-window-navigator-trong-javascript-389.html#goto-h2-1)
* [3. Kiểm tra Engine của trình duyệt](http://freetuts.net/bom-window-navigator-trong-javascript-389.html#goto-h2-2)
* [4. Kiểm tra Vesion của trình duyệt](http://freetuts.net/bom-window-navigator-trong-javascript-389.html#goto-h2-3)
* [5. Kiểm tra hệ điều hành của Client](http://freetuts.net/bom-window-navigator-trong-javascript-389.html#goto-h2-4)
* [6. Kiểm tra ngôn ngữ của trình duyệt](http://freetuts.net/bom-window-navigator-trong-javascript-389.html#goto-h2-5)
* [Lời kết](http://freetuts.net/bom-window-navigator-trong-javascript-389.html#goto-h2-6)

Đối tượng **Window Navigator trong Javascript** được dùng để kiểm tra các thông tin về người dùng như trình duyệt đang sư dụng là gì? hệ điều hành đang sử dụng là gì? Trình duyệt có bật[Cookie](http://freetuts.net/bom-cookie-trong-javascript-388.html) hay không? hay thậm chí có thể kiểm tra được tên và version của Browser.

Trong bài học chúng ta sẽ tìm hiểu các thành phần chính liên quan tới Window Navigator như sau:

* navigator.cookieEnabled
* navigator.appName và navigator.appCodeName
* navigator.product
* navigator.appVersion và navigator.userAgent
* navigator.platform
* navigator.language

**1. Kiểm tra Cookie có được bật không?**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Để kiểm tra trình duyệt có bật Cookie hay không thì ta sử dụng thuộc tính navigator.cookieEnabled.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=329)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (window.cookieEnabled){      alert("Có bật Cookie - freetuts.net");  }  else{      alert("Cookie đã bị tắt");  } |

**2. Kiểm tra tên trình duyệt đang sử dụng**

Để kiểm tra tên trình duyệt thì ta dùng thuộc tính navigator.appName và thuộc tính navigator.appCodeName dùng để kiểm tra tên mã code của trình duyệt.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=330)

|  |  |
| --- | --- |
| 1  2 | document.write("App Name: " + window.navigator.appName + "<br/>");  document.write("Code Name: " + window.navigator.appCodeName); |

**3. Kiểm tra Engine của trình duyệt**

Để kiểm tra Engine của trình duyệt ta sử dụng thuộc tính navigator.product.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=331)

|  |  |
| --- | --- |
| 1 | document.write("Engine: " + navigator.product); |

**4. Kiểm tra Vesion của trình duyệt**

Để kiểm tra Version của trình duyệt ta sử dụng thuộc tính navigator.appVersion hoặc navigator.userAgent.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=332)

|  |  |
| --- | --- |
| 1  2 | document.write("Cách 1: " + navigator.appVersion + "<br/>");  document.write("Cách 1: " + navigator.userAgent); |

Tuy nhiên việc sử dụng Javascript để kiểm tra Version sẽ có lúc kết quả trả về sai, vì vậy bạn không nên sử dụng nó để kiểm tra Version của trình duyệt.

**5. Kiểm tra hệ điều hành của Client**

Javascript cũng có thể xem hệ điều hành mà người dùng đang sử dụng bằng cách sử dụng thuộc tính navigator.platform.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=333)

|  |  |
| --- | --- |
| 1 | document.write("Hệ điều hành: " + navigator.platform); |

**6. Kiểm tra ngôn ngữ của trình duyệt**

Mỗi trình duyệt có thể lựa chọn ngôn ngữ khác nhau và ta có thể kiểm tra bằng phương thức navigator.language.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=334)

|  |  |
| --- | --- |
| 1 | document.write("Ngôn ngữ Browser: " + navigator.language); |

**Lời kết**

Việc sử dụng đối tượng Navigator này bạn có thể lấy toàn bộ thông tin về trình duyệt và hệ điều hành của người dùng, và nếu bạn muốn lấy các thông tin đó lưu vào cơ sở dữ liệu thì hãy sử dụng [Ajax](http://freetuts.net/tag/ajax-toan-tap), tuy nhiên trong thực tế ta rất ít khi sử dụng các thông tin đó.

**Bài 35: BOM - Screen trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 08-05-2015- View: 2517

**Mục luc:**

* [1. Lấy width và height của màn hình](http://freetuts.net/bom-screen-trong-javascript-390.html#goto-h2-0)
* [2. lấy Color Depth của screen](http://freetuts.net/bom-screen-trong-javascript-390.html#goto-h2-1)
* [3. Lấy số Pixel depth của screen](http://freetuts.net/bom-screen-trong-javascript-390.html#goto-h2-2)
* [Lời kết](http://freetuts.net/bom-screen-trong-javascript-390.html#goto-h2-3)

Tiếp theo loạt bài tìm hiểu [BOM trong Javascript](http://freetuts.net/bom-la-gi-bom-trong-javascript-384.html) thì trong bài này chúng ta tìm hiểu đối tượng Window Screen, đây là đối tượng chứa thông tin về màn hình của trình duyệt người dùng.

Chúng ta tìm hiểu một số thuộc tính sau:

* screen.width
* screen.height
* screen.availWidth
* screen.availHeight
* screen.colorDepth
* screen.pixelDepth

**Note**: Khi chúng ta ghi screen.with tức là ghi tắt của window.screen.with.

**1. Lấy width và height của màn hình**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Chúng ta sử dụng thuộc tính screen.width để lấy chiều rộng và screen.height để lấy chiều cao của màn hình, kết quả sẽ trả về định dạng Pixels.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=335)

|  |  |
| --- | --- |
| 1  2 | document.write("With screen: " + screen.width + "<br/>");  document.write("Height screen: " + screen.height); |

Ngoài ra để lấy chiều rộng và chiều cao mà không tính các tools tính năng của trình duyệt thì bạn sử dụng screen.availWidth và screen.availHeight.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=336)

|  |  |
| --- | --- |
| 1  2 | document.write("With Available screen: " + screen.availWidth + "<br/>");  document.write("Height Available screen: " + screen.availHeight); |

**2. lấy Color Depth của screen**

Để lấy **color depth** của screen thì ta sử dụng thuộc tính screen.colorDepth.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=337)

|  |  |
| --- | --- |
| 1 | document.write("Color Depth: " + screen.colorDepth); |

**3. Lấy số Pixel depth của screen**

Để lấy **Pixel depth** của màn hình thì ta sử dụng thuộc tính screen.pixelDepth.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=338)

|  |  |
| --- | --- |
| 1 | document.write("Pixel Depth: " + screen.pixelDepth); |

**Lời kết**

Thực tế những thông tin về screen này ta rất ít khi sử dụng khi xây dựng website mà chỉ dùng khi bạn lập trình Game bằng javascript, vì vậy nếu bạn là Website Developer thì có thể không cần phải xem bài này.

**Bài 36: setTimeout() và setInterval() trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 09-05-2015- View: 9538

**Mục luc:**

* [1. Hàm setTimeout() trong Javascript](http://freetuts.net/settimeout-va-setinterval-trong-javascript-391.html#goto-h2-0)
  + [Hàm clearTimeout() trong Javascript](http://freetuts.net/settimeout-va-setinterval-trong-javascript-391.html#goto-h3-0)
* [2. Hàm setInterval() trong Javascript](http://freetuts.net/settimeout-va-setinterval-trong-javascript-391.html#goto-h2-1)
  + [hàm clearInterval() trong Javascript](http://freetuts.net/settimeout-va-setinterval-trong-javascript-391.html#goto-h3-1)
* [3. Lời kết](http://freetuts.net/settimeout-va-setinterval-trong-javascript-391.html#goto-h2-2)

Trong bài này chúng ta tìm hiểu hai hàm thường dùng để xử lý thời gian đó là setTimeout() vàsetInterval(), hai hàm này có chung một đặc điểm là thiết lập và thực hiện một nhiệm vụ nào đó trong một khoảng thời gian nào đó, số lần thực hiện có thể được lặp đi lặp lại hoặc không. Tuy vậy giữa hai hàm cũng có sự khác nhau về **số lần lặp** và chi tiết thế nào thì các bạn xem các phần dưới đây.

**1. Hàm setTimeout() trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Hàm setTimeout() dùng để thiết lập một khoảng thời gian nào đó sẽ thực hiện một nhiệm vụ nào đó và nó chỉ thực **hiện đúng một lần**.

**Cú pháp**: setTimeout(function, time)

**Trong đó**:

* function: là nội dung cần thực hiện, đây là một [hàm](http://freetuts.net/ham-va-tao-ham-function-trong-javascript-274.html)
* time: là khoảng thời gian bao nhiêu (*tính bằng mili giây*) thì function đó sẽ thực hiện

**Ví dụ**: Sau 3 giây thì xuất hiện câu chào lên màn hình

[XEM DEMO](http://freetuts.net/editor.html?id=339)

|  |  |
| --- | --- |
| 1  2  3 | setTimeout(function(){      alert("Chào mừng bạn đến với freetuts.net");  }, 3000); |

Trong ví dụ này mình đã truyền tham số thứ nhất là một function và tham số thứ hai là số giây mà function sẽ được thực hiện. Ngoài ra bạn có thể viết lại như sau cho dễ nhìn:

[XEM DEMO](http://freetuts.net/editor.html?id=340)

|  |  |
| --- | --- |
| 1  2  3  4 | var do\_alert = function(){      alert("Chào mừng bạn đến với freetuts.net");  };  setTimeout(do\_alert, 3000); |

**Hàm clearTimeout() trong Javascript**

Giả sử bạn xây dựng chức năng sau 3 giây thì sẽ xuất hiện thông báo, tuy nhiên sau 2 giây chương trình muốn hủy bỏ thì phải làm thế nào? Lúc này bạn phải sử dụng hàm clearTimeout().

Tham số truyền vào hàm clearTimeout() là đối tượng setTimeout() nên lúc này ban phải đặt hàm setTimeout() vào một biến cụ thể.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // hành động  var action = setTimeout(function(){      // something  }, 3000);    // hủy hành động  clearTimeout(action); |

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=341)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <script language="javascript">      var do\_alert = setTimeout(function(){          alert("Chào mừng bạn đến với freetuts.net");      }, 3000);        function clearAlert()      {          clearTimeout(do\_alert);      }  </script>    <input type="button" onclick="clearAlert()" value="Clear" /> |

**2. Hàm setInterval() trong Javascript**

Hàm setInterval() có cú pháp và chức năng giống như hàm setTimeout(), tuy nhiên với hàm setInterval() thì **số lần thực hiện lã mãi mãi**.

**Ví dụ**: Cứ sau 3 giây thì xuất hiện câu chào một lần

[XEM DEMO](http://freetuts.net/editor.html?id=342)

|  |  |
| --- | --- |
| 1  2  3 | setInterval(function(){      alert("Chào mừng bạn đến với freetuts.net");  }, 3000); |

Trong ví dụ này cứ sau ba giây thì sẽ xuất hiện câu chào và số lần thực hiện sẽ là vĩnh viễn, không tin bạn cứ chạy trang web lên và để vậy.

**hàm clearInterval() trong Javascript**

Tương tự như hàm clearTimeout(), hàm clearInterval() sẽ xóa đi nhiệm vụ mà ta đã thiết lập trong hàm setInterval(), và ta cũng phải đặt setInterval() trong một biến thì mới clear được.

**Ví dụ**: Sử dụng hàm setInterval() để xuất câu chào lên màn hình và số lần xuất hiện chỉ 1 lần duy nhất

[XEM DEMO](http://freetuts.net/editor.html?id=343)

|  |  |
| --- | --- |
| 1  2  3  4 | var interval\_obj = setInterval(function(){      alert("Chào mừng bạn đến với freetuts.net");      clearInterval(interval\_obj);  }, 3000); |

Trong ví dụ này nội dung thực hiện là xuất hiện câu chào và **ngay lập tức xóa nhiệm vụ của interval** luôn nên nó chỉ thực hiện 1 lần.

**3. Lời kết**

Như vậy sự khác nhau giữa hàm setTimeout() và setInterval() là đối với hàm setTimeout() thì số lần thực hiện là 1 lần và đối với setInterval() thì số lần thực hiện sẽ là mãi mãi cho tới khi bạn sử dụng hàm clearInterval() để can thiệp vào. Việc sử dụng hàm nào thì phụ thuộc vào nhu cầu của bạn và hãy lưu ý điểm khác nhau này để làm cho đúng.

Bài này mình sẽ dừng ở đây, hy vọng  qua các ví dụ bạn sẽ hiểu được cách sử dụng hàm setTimeout() và setInterval() trong Javascript, chúc bạn học tốt.

**Bài 37: Chuỗi (string) trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 09-05-2015- View: 6217

**Mục luc:**

* [1. Chuỗi (String) trong Javascript](http://freetuts.net/chuoi-string-trong-javascript-392.html#goto-h2-0)
* [2. Nối chuỗi trong Javascript](http://freetuts.net/chuoi-string-trong-javascript-392.html#goto-h2-1)
  + [Lưu ý khi xuống hàng của chuỗi trong Javascript](http://freetuts.net/chuoi-string-trong-javascript-392.html#goto-h3-0)
* [3. Ép sang kiểu chuỗi String trong javascript](http://freetuts.net/chuoi-string-trong-javascript-392.html#goto-h2-2)
* [4. Lời kết](http://freetuts.net/chuoi-string-trong-javascript-392.html#goto-h2-3)

Như vậy là ta đã tìm hiểu các kiến thức riêng của javascript như [DOM](http://freetuts.net/dom-la-gi-cac-loai-dom-trong-javascript-366.html), [BOM](http://freetuts.net/bom-la-gi-bom-trong-javascript-384.html), [Event](http://freetuts.net/su-kien-event-trong-javascript-368.html) nên để tiếp tục serie thì ta quay lại học kỹ thuật lập trình javascript căn bản, trong bài này chúng ta tìm hiểu đến chuỗi (string) trong Javascript.

**1. Chuỗi (String) trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Ở các bài trước chúng ta đã thực hiện khá nhiều ví dụ có sử dụng chuỗi nhưng chưa tìm hiểu về nó nên bây giờ ta tìm hiểu định nghĩa chính xác.

Chuỗi là một đoạn text có thể có một hoặc nhiều ký tự và thông thường chúng ta sẽ lưu trữ nó vào một biến, biến này ta sẽ gọi là biến có kiểu dữ liệu là String (chuôi). Tất cả các chuỗi đều phải được bao quanh bằng cặp dấu nháy đơn ' hoặc nháy đôi ".

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2 | var website = "website freetuts.net học javascript quá sướng";  var email = 'thehalfheart@gmail.com';<br> |

Trường hợp trong chuỗi **cũng có xuất hiện** dấu nháy đơn hoặc nháy đôi thì bắt buộc bạn phải thêm ký tự \ đằng trước dấu nháy đó nếu không sẽ bị lỗi cũ pháp.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2 | var message = "Học lập trình tại \"freetuts.net\" quá sướng";  var domain = 'freetuts.net - \'web học lập trình\''; |

Ngoài ra còn nhiều ký hiệu kết hợp với dấu \ nữa như trong bảng dưới đây:

| **Code** | **Kết quả** |
| --- | --- |
| \' | single quote |
| \" | double quote |
| \\ | backslash |
| \n | new line |
| \r | carriage return |
| \t | tab |
| \b | backspace |
| \f | form feed |

**2. Nối chuỗi trong Javascript**

Để nối chuỗi chúng ta sử dụng dấu + để gép hai chuỗi (hoặc biến) lại với nhau.

**Ví dụ**:

|  |  |
| --- | --- |
| 1 | var message = "chào mừng bạn" + "đến với freetuts.net"; |

Hoặc:

|  |  |
| --- | --- |
| 1  2  3  4  5 | var message1 = "chào mừng bạn";  var message2 = "đến với freetuts.net";    // Nối hai chuỗi  var message = message1 + message2; |

**Lưu ý khi xuống hàng của chuỗi trong Javascript**

Khi bạn muốn Enter xuống hàng một chuỗi trong Javascript thì bắt buộc phải sử dụng dấu +để nối chuỗi nếu không sẽ bị lỗi cú pháp.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // Đúng  var message = "Chào mừng bạn đến với"                + "freetuts.net";    // Sai  var message = "Chào mừng bạn đến với                 freetuts.net"; |

Nếu bạn muốn viết gọn hơn thì sử dụng dấu \ để báo cho trình duyệt biết là có xuống hàng.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2 | var message = "Chào mừng bạn đến với \                           freetuts.net";<br /> |

**3. Ép sang kiểu chuỗi String trong javascript**

Nếu bạn muốn ép một giá trị nào đó sang kiểu chuỗi thì có thể sử dụng cú pháp string.toString().

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=344)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // Trước khi chuyển đổi  var number = 12;  alert(typeof number);    // Sau khi chuyển đổi  number = number.toString();  alert(typeof number); |

Từ khóa typeof vars sẽ trả về kiểu dữ liệu của biến vars.

Ngoài cách trên bạn có thể sử dụng đối tượng String để tạo một chuỗi và vì là đối tượng nên nó phải có từ khóa new đặt đằng trước.

**Ví dụ**:

|  |  |
| --- | --- |
| 1 | var message = new String("chào mừng bạn đến với freetuts.net"); |

Lúc này dù bạn truyền vào là kiểu gì đi nữa thì kết quả nó vẫn trả về kiểu string, tuy nhiên người ta khuyến khích bạn **không nên** sử dụng cách này bởi vì chương trình sẽ chạy chậm hơn.

**4. Lời kết**

Việc xử lý chuỗi trong Javascript cũng tương đối quan trọng và thường xuyên sử dụng nên bạn cần lưu ý các phần trên để tránh mắc phải những lỗi không đáng có. Và mình cũng nhắc lại việc học Javascript bạn nên [sử dụng Firebug để kiểm tra lỗi](http://freetuts.net/su-dung-firebug-de-kiem-tra-loi-javascript-380.html) bởi vì nó trực quan và đầy đủ nhất.

Bài này sẽ kết thúc ở đây, bài tiếp theo chúng ta sẽ tìm hiểu các hàm xử lý chuỗi trong Javascript.

**Bài 38: Các hàm xử lý chuỗi trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 09-05-2015- View: 21493

**Mục luc:**

* [1. Các hàm xử lý chuỗi trong Javascript](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h2-0)
  + [Tìm kiếm chuỗi con](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h3-0)
  + [Cắt chuỗi con](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h3-1)
  + [Tìm kiếm và lặp chuỗi](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h3-2)
  + [Chuyển thành chữ hoa và chữ thường](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h3-3)
  + [Nối thêm chuỗi](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h3-4)
  + [Tìm ký tự hoặc mã ASCII của một ký tự](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h3-5)
  + [Chuyển đổi chuỗi sang mảng](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h3-6)
* [2. Lời kết](http://freetuts.net/cac-ham-xu-ly-chuoi-trong-javascript-393.html#goto-h2-1)

Bài trước chúng ta đã tìm hiểu về [chuỗi trong javascript](http://freetuts.net/chuoi-string-trong-javascript-392.html) nhưng đó chỉ là những thao tác đơn giản như khởi tạo, gán giá trị và thực hiện toán tử nối chuỗi, vậy trong bài này mình sẽ giới thiệu những hàm xử lý chuỗi trong Javascript thường sử dụng nhất để các bạn sử dụng trong quá trình làm việc. Nói là hàm nhưng thực chất đó là những phương thức vì tất cả những hàm đó đều phải sử dụng thông qua đối tượng String và chi tiết thế nào thì chúng ta cùng tìm hiểu nhé.

**1. Các hàm xử lý chuỗi trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Mình không thể liệt kê hết các hàm được mà chỉ đưa ra những hàm hay sử dụng nhất để bài viết được ngắn gọn hơn.

**Tìm kiếm chuỗi con**

Chúng ta có ba hàm thường dùng để tìm kiếm chuỗi con trong Javascript như sau:

* indexOf()
* lastIndexOf()
* search()

**Hàm indexOf()**

Để tìm kiếm chuỗi con thì ta sử dụng hàm String.indexOf(str), trong đó str là chuỗi con và String là chuỗi cha. Hàm này sẽ trả kết quả về kết quả là vị trí xuât hiện đầu tiên của chuỗi (bắt đầu là vị trí 0), nếu không tìm thấy chuỗi con thì nó sẽ trả về -1.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=351)

|  |  |
| --- | --- |
| 1  2 | var string = "Chào mừng bạn đến với freetuts.net";  document.write("Vị trí xuất hiện chuỗi freetuts.net là: " + string.indexOf("freetuts.net"));<br> |

**Hàm lastIndexOf()**

Trường hợp nếu chuỗi con xuất hiện nhiều lần trong chuỗi cha thì kết quả cũng trả về vị trí xuất hiện của chuỗi con đầu tiên. Vậy làm thế nào để lấy vị trí của chuỗi con cuối cùng trong chuỗi cha? Ta sẽ sử dụng hàm String.lastIndexOf(str), hàm này sẽ trả về vị trí xuất hiện của chuỗi con cuối cùng và trả về -1 nếu không tìm thấy.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=352)

|  |  |
| --- | --- |
| 1  2 | var string = "Website freetuts.net - học lập trình miễn phí tại freetuts.net";  document.write("Vị trí xuất hiện chuỗi freetuts.net là: " + string.lastIndexOf("freetuts.net")); |

**Hàm search()**

Ngoài hai hàm trên bạn có thể sử dụng hàm string.search(str) để tìm kiếm, tác dụng của nó cũng giống như hàm string.indexOf(str).

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=353)

|  |  |
| --- | --- |
| 1  2 | var string = "Chào mừng bạn đến với freetuts.net";  document.write("Vị trí xuất hiện chuỗi freetuts.net là: " + string.search("freetuts.net"));<br> |

**Cắt chuỗi con**

Nếu ban muốn cắt một chuỗi con từ chuỗi cha thì bạn có thể sử dụng ba hàm sau:

* slice(start, end)
* substring(start, end)
* substr(start, length)

**Note**: tất cả các vị trí của chuỗi đều bắt đầu từ 0, vì vậy khi tính toán vị trí coi chừng bị nhầm lẫn nhé.

**Hàm slice()**

Hàm slide có hai tham số truyền vào:

* start: vị trí bắt đầu
* end: vị trí kết thúc

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=354)

|  |  |
| --- | --- |
| 1  2 | var string = "Welcome to freetuts.net";  document.write("Chuỗi cần lấy là: " + string.slice(11, 23));<br> |

Nếu tham số truyền vào là số âm thì nó sẽ tính ngược lại, nghĩa là nó sẽ đếm từ cuối lên.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=355)

|  |  |
| --- | --- |
| 1  2 | var string = "Welcome to freetuts.net";  document.write("Chuỗi cần lấy là: " + string.slice(-12, 23)); |

Nếu bạn chỉ truyền một tham số đầu tiên thì nó sẽ tự hiểu vị trí end là vị trí cuối cùng.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=356)

|  |  |
| --- | --- |
| 1  2 | var string = "Welcome to freetuts.net";  document.write("Chuỗi cần lấy là: " + string.slice(5)); |

**Hàm substring()**

Hàm substring() có cách sử dụng giống với hàm slice(), tuy nhiên tham số truyền vào hàm substring() phải luôn luôn lớn hơn 0.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=357)

|  |  |
| --- | --- |
| 1  2 | var string = "Welcome to freetuts.net";  document.write("Chuỗi cần lấy là: " + string.substring(11, 23)); |

**Hàm substr()**

Hàm substr() có hai tham số là start và length, trong đó start là vị trí bắt đầu và length là số ký tự muốn lấy bắt đầu từ vị trí start. Nếu bạn truyền tham số start là số âm thì nó sẽ tính từ cuối trở lên, còn tham số length phải luôn luôn là số dương.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=358)

|  |  |
| --- | --- |
| 1  2 | var string = "Welcome to freetuts.net";  document.write("Chuỗi cần lấy là: " + string.substr(11, 12)); |

**Tìm kiếm và lặp chuỗi**

Để tìm kiếm và lặp một chuỗi con nào đó thì bạn sử dụng hàm replace(str\_find, str\_replace), trong đó str\_find là chuỗi cần tìm và str\_replace là chuỗi sẽ được thay thế chuỗi str\_find.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=359)

|  |  |
| --- | --- |
| 1  2 | var string = "Welcome to freetuts.net";  document.write(string.replace("freetuts.net", "freetuts.net")); |

**Chuyển thành chữ hoa và chữ thường**

Để chuyển chuỗi thành chữ hoa ta dùng hàm toUpperCase() và chuyển thành chữ thường ta dùng hàm toLowerCase().

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=360)

|  |  |
| --- | --- |
| 1  2  3 | var string = "Welcome to freetuts.net";  document.write(string.toUpperCase() + "<br/>");  document.write(string.toLowerCase()); |

**Nối thêm chuỗi**

Để nối thêm chuỗi thông thường ta dùng toán tử +, ngoài ra bạn có thể dùng hàm concat()để thực hiện nối chuỗi.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=361)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var string = "Welcome " + "to" + " freetuts.net";  document.write(string + "<br/>");    // hoặc  var string = "Welcome ";  string = string.concat("to ", "freetuts.net");  document.write(string + "<br/>"); |

**Tìm ký tự hoặc mã ASCII của một ký tự**

Để xem ký tự của một vị trí nào đó thì dùng hàm charAt(), còn xem [mã ASCII](http://freetuts.net/ma-ascii-la-gi-bang-ma-ascii-358.html) thì dùng hàm charCodeAt(). Cả hai hàm này đều có tham số truyền vào là vị trí muốn xem.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=362)

|  |  |
| --- | --- |
| 1  2  3  4 | var string = "Welcome freetuts.net";    document.write(string.charAt(1) + "<br/>");  document.write(string.charCodeAt(1) + "<br/>"); |

**Chuyển đổi chuỗi sang mảng**

Để chuyển một chuỗi sang mảng thì ta sử dụng hàm split() với tham số truyền vào là ký tự ngăn cách giữa các phần tử.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=363)

|  |  |
| --- | --- |
| 1  2  3  4  5 | string = "Welcome freetuts.net";    // Tạo thành mảng với mỗi phần tử ngăn bởi khoảng trắng  // kết quả là mảng có hai phần tử gồm: welcome và feetuts.net  document.write(string.split(" ").length); |

**2. Lời kết**

Trên là các hàm xử lý chuỗi thông dụng mà bạn nên biết để sử dụng khi cần thiết, các hàm này đang ở mức căn bản vì vẫn chưa đáp ứng được các yêu cầu xử lý phức tạp hơn, vấn đề này mình sẽ đề cập sau khi chúng ta tìm hiểu về Regex trong Javascript.

Nếu bạn cần thêm bài tập để thực hành thì có thể theo dõi ở serie bài tập Javascript, nơi này là tổng hợp các [bài tập Javascript](http://freetuts.net/tag/bai-tap-javascript) từ căn bản đến nâng cao.

**Bài 39: Hàm typeof trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 18-05-2015- View: 3971

**Mục luc:**

* [1. Hàm typeof trong Javascript](http://freetuts.net/ham-typeof-trong-javascript-398.html#goto-h2-0)
* [2. Lời kết](http://freetuts.net/ham-typeof-trong-javascript-398.html#goto-h2-1)

Mình gọi là hàm typeof nhưng thực chất không biết có phải là hàm hay không vì nó không giống một hàm mà giống một từ khóa thì đúng hơn, tuy nhiên để tiện cho việc học và viết tutorial thì mình sẽ gọi là hàm typeof nhé.

**1. Hàm typeof trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Hàm typeof được dùng để kiểm tra một biến nào đó (*hoặc một giá trị nào đó*) có kiểu dữ liệu là gì? Cú pháp của nó khá đơn giản như sau:

|  |  |
| --- | --- |
| 1 | var x = typeof value; |

Trong đó value có thể là một biến hoặc một giá trị xác định.

Thông thường chúng ta có một số loại dữ liệu như sau:

* number - số
* string - chuỗi
* object - đối tượng
* undefined - không xác định

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=368)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var number = 12;  var string = "12";  var object = new Number();    document.write("number: " + typeof number + "<br/>");  document.write("string: " + typeof string + "<br/>");  document.write("object: " + typeof object + "<br/>"); |

Trong PHP để kiểm tra một biến nào đó có tồn tại hay không thì ta dùng hàm [isset()](http://freetuts.net/ham-isset-va-empty-trong-php-350.html), còn trong Javascript thì ta dùng hàm typeof kết hợp với kiểu dữ liệu undefined, nghĩa là nếu kiểu dữ liệu của một biến nào đó là undefined thì tức là biến đó chưa được định nghĩa hoặc giá trị của nó không xác định.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=369)

|  |  |
| --- | --- |
| 1  2  3 | if (typeof variable == 'undefined'){      document.write("Biến variable không được định nghĩa");  } |

**2. Lời kết**

Trên là một số vấn đề liên quan đến hàm typeof trong Javascript, mục đích mình viết bài này là để các bài sau sử dụng lại hàm này và mình không mất công giải thích nữa. Còn một điều nữa là hàm typeof thường được dùng để kiểm tra trình duyệt có hỗ trợ hàm nào đó hay không.

**Bài 40: Đối tượng Number trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 19-05-2015- View: 2866

**Mục luc:**

* [1. Đối tượng Number trong Javascript](http://freetuts.net/doi-tuong-number-trong-javascript-397.html#goto-h2-0)
* [2. Xử lý Number trong Javascript](http://freetuts.net/doi-tuong-number-trong-javascript-397.html#goto-h2-1)
  + [Chuyển Number sang String](http://freetuts.net/doi-tuong-number-trong-javascript-397.html#goto-h3-0)
  + [Số Infinity](http://freetuts.net/doi-tuong-number-trong-javascript-397.html#goto-h3-1)
  + [NaN - Not a Number](http://freetuts.net/doi-tuong-number-trong-javascript-397.html#goto-h3-2)
  + [Numbers - Objects](http://freetuts.net/doi-tuong-number-trong-javascript-397.html#goto-h3-3)
* [3. Lời kết](http://freetuts.net/doi-tuong-number-trong-javascript-397.html#goto-h2-2)

Trong Javascript có một đối tượng thường được dùng để xử lý kiểu dữ liệu number đó là đối tượng Number, đối tượng này thường có hai dạng đó là số có dấu chấm động và số không có dấu chấm động.

Trong bài này chúng ta sẽ tìm hiểu cách khai báo Number và một số vấn đề liên quan đến đối tượng Number này.

**1. Đối tượng Number trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Trong Javascript có hai giá trị lưu trữ trong Number đó là số có dấu chấm động và không có dấu chấm động.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2 | var x = 12; // không có dấu chấm động  var y = 12.5; // có dấu chấm động |

Và nếu một số quá lớn thì bạn có thể dùng số mũ để biểu diễn.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2 | var x = 123e5;    // 12300000  var y = 123e-5;   // 0.00123 |

Không giống với các ngôn ngữ lập trình khác, trong Javascript các số luôn luôn là 64 bít và kiểu float, vì vậy bạn không thể định nghĩa các kiểu dữ liệu như integer, short, long, ... hay nói cach khác trong Javascript khi làm việc với các chữ số thì chỉ có khái niệm Number.

Ngoài ra bạn có thể biểu diễn giá trị của Number ở dạng nhị phân, thập phân, thập lục phân, ...

**Ví dụ**:

|  |  |
| --- | --- |
| 1 | var x = 0xFF;             // x có giá trị bằng 255 |

Trong Javascript tất cả các dữ liệu liên quan đến những con số đều có kiểu dữ liệu là Number, nghĩa là bạn dùng hàm [typeof](http://freetuts.net/ham-typeof-trong-javascript-398.html) thì sẽ có kết quả trả về là '*number*'.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var x = 123;  var y = 1.23;    typeof 12;  // number  typeof x;   // number  typeof y;   // number |

**2. Xử lý Number trong Javascript**

Phần này chúng ta tìm hiểu một số cách xử lý liên quan đến đối tượng Number và mình cũng đưa ra một số lưu ý để bạn không bị mắc lỗi khi sử dụng nó.

**Chuyển Number sang String**

Để chuyển một biến đang ở kiểu Number sang kiểu String thì ta sử dụng phương thức number.toString(type), hàm này có một tham số truyền vào là  type và đây chính là kiểu dữ liệu mà ban muốn chuyển về, mặc định sẽ là hệ thập phân (10).

Sau đây là danh sách các hệ cơ số thông dụng:

* Hệ nhị phân (2)
* Hệ bát phân (8)
* Hệ thập phân (10)
* Hệ thập lục phân (8)

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4 | var myNumber = 128;  myNumber.toString(16);    // returns 80  myNumber.toString(8);      // returns 200  myNumber.toString(2);      // returns 10000000 |

**Số Infinity**

Infinity cũng là một kiểu dữ liệu Number và khi một biến có giá trị là Infinity thì tức là nó đã vượt mức lưu trữ cho phép nên theo mặc định nó sẽ chuyển về dạng đó. Vì nó cũng là một giá trị nên bạn có thể so sánh bình thường.

**Ví dụ**: Ta sử dụng [vòng lặp while](http://freetuts.net/vong-lap-while-do-while-trong-javascript-271.html) để lặp cho tới khi biến myNumber có giá trị là Infinity.

[XEM DEMO](http://freetuts.net/editor.html?id=366)

|  |  |
| --- | --- |
| 1  2  3  4  5 | var myNumber = 2;  while (myNumber != Infinity) {      myNumber = myNumber \* myNumber;  }  document.write("Giá trị của myNumber là: " + myNumber); |

**NaN - Not a Number**

Nếu bạn thực hiện một phép toán nào đó liên quan đến Number nhưng vi phạm quy tắc tính toán thì kết quả sẽ trả về một giá trị gọi là NaN (*Not a Number*). Ví dụ khi bạn thực hiện phép chia hai số nếu bạn cho mẫu số là một chuỗi String thì kết quả sẽ là NaN.

**Ví dụ**:

|  |  |
| --- | --- |
| 1 | var x = 2/"freetuts.net"; // NaN |

Tuy nhiên nếu mẫu số là một chuỗi Number thì kết quả nó vẫn hoạt động bình thường.

**Ví dụ**:

|  |  |
| --- | --- |
| 1 | var x = 100 / "10"; // kết quả = 10 |

Để kiểm tra một biến nào đó có phải là NaN hay không thì chúng ta sử dụng hàm isNaN().

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=367)

|  |  |
| --- | --- |
| 1  2  3  4 | var x = 2/"s";  if (isNaN(x)){      document.write("NaN");  } |

**Numbers - Objects**

Ngoài cách tạo một biến Number thông thường đó là gán giá trị trực tiếp thì còn một cách khác đó là sử dụng đối tượng Number. Tuy nhiên hai cách tạo này sẽ có kiểu dữ liệu khác nhau.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2 | var x = 123; // number  var y = new Number(123); // object |

**3. Lời kết**

Vấn đề quan trọng nhất của bài này là hiểu rõ bản chất của Number, cách kiểm tra một biến có phải là "*Not a Number*" hay không bằng cách sử dụng hàm isNaN(), khái niệm Infinity là gì?

Ngoài các cách sử dụng trên vẫn còn nhiều vấn đề nữa nhưng mình không thể trình bày hết được. Bài tiếp theo chúng ta tìm hiểu một số hàm xử **Number trong Javascript**.

**Bài 41: Hàm xử lý Number trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 19-05-2015- View: 7990

**Mục luc:**

* [1. Hàm xử lý Number toàn cục](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h2-0)
  + [Number()](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h3-0)
  + [parseInt()](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h3-1)
  + [parseFloat()](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h3-2)
* [2. Hàm xử lý Number cục bộ](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h2-1)
  + [toString()](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h3-3)
  + [toFixed(n)](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h3-4)
  + [toPrecision(n)](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h3-5)
  + [valueOf()](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h3-6)
* [3. Lời kết](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html#goto-h2-2)

Trong bài tìm hiểu đối tượng [Number trong Javascript](http://freetuts.net/doi-tuong-number-trong-javascript-397.html) mình đã giới thiệu với các bạn cách hoạt động và một số lưu ý, vậy thì trong bài này mình sẽ tiếp tục viết về Number và mình sẽ giới thiệu các hàm thường được dùng để xử lý Number trong Javascript.

Các hàm xử lý mình chia làm hai nhóm chính, thứ nhất là nhóm toàn cục và thứ hai là nhóm cục bộ, nhóm toàn cục tức là những hàm không nằm trong đối tượng Number và nhóm cục bộ tức là những hàm nằm trong đối tượng Number.

**Nhóm toàn cục**:

* Number(type) trả về một number và định dạng cơ số type = (*nhị phân, thập phân, thập lục phân*)
* parseFloat() chuyển sang một số float
* parseInt() chuyển sang một số integer

**Nhóm cục bộ**:

* toString() chuyển sang kiểu  string
* toFixed(n) chuyển sang số có n số lẻ ở đằng sau
* toPrecision(n) chuyển sang số có chiều dài là n
* valueOf() lấy giá trị của biến hoặc một giá trị nào đó

**1. Hàm xử lý Number toàn cục**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Chúng ta sẽ lần lượt tìm hiểu ba hàm toàn cục trên.

**Number()**

Dùng để chuyển đổi một biến hoặc một giá trị nào đó sang kiểu number, nó chuyển tất cả các định dạng như Boolean, Date, String. Nếu trường hợp giá trị cần chuyển đổi không thể chuyển sang Number được thì nó sẽ chuyển sang giá trị mặc định là NaN.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=370)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | var boolean\_true = true;  Number(boolean\_true);           // returns 1    var boolean\_false = false;  Number(boolean\_false);          // returns 0    var string\_str = 'freetuts.net';  Number(string\_str);             // returns NaN    var string\_num = '100';  Number(boolstring\_numean\_true); // returns 100    var date = new Date();  Number(boolean\_true);           // returns 1 |

**parseInt()**

Hàm này có tác dụng giống như hàm Number(), tuy nhiên có một số điểm khác biệt như sau:

* Nếu chuỗi có các ký tự đầu tiên là các con số và ở đằng sau là chữ cái thì nó sẽ lấy các số đầu tiên đó và chuyển thành kiểu number. Trường hợp này nếu dùng hàm Number()thì nó sẽ chuyển thành NaN.
* Nếu dữ liệu ở các định dạng khác string thì nó sẽ chuyển thành NaN

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=371)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | var boolean\_true = true;  parseInt(boolean\_true); // returns NaN    var boolean\_false = false;  parseInt(boolean\_false);// returns NaN    var string\_str = '10 freetuts.net';  parseInt(string\_str);   // returns 10    var string\_num = '100';  parseInt(string\_num);   // returns 100    var date = new Date();  parseInt(boolean\_true); // returns NaN |

**parseFloat()**

Hàm này chuyển dữ liệu sang định dạng float, về cách sử dụng nó giống với hàm parseInt().

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=372)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | var boolean\_true = true;  parseFloat(boolean\_true);   // returns NaN    var boolean\_false = false;  parseFloat(boolean\_false);  // returns NaN    var string\_str = '10.2 freetuts.net';  parseFloat(string\_str);     // returns 10.2    var string\_num = '100';  parseFloat(string\_num);     // returns 100    var date = new Date();  parseFloat(boolean\_true);   // returns NaN |

**2. Hàm xử lý Number cục bộ**

Những hàm cục bộ phải gắn liền với đối tượng Number cụ thể, ví dụ khi bạn khởi tạo một biến var x = 12 thì lúc này x có tất cả các hàm cục bộ đó. Ngoài ra bạn có thể sử dụng cặp mở đóng ()  để bao quanh một biểu thức  hoặc một giá trị thì vẫn sử dụng bình thường.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4 | var x = 12;  x.toString();  (12).toString();  (12 + 12).toString(); |

**toString()**

Hàm toString() có tác dụng chuyển đổi Number sang kiểu String.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=373)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var x = 123;    typeof x;   // number    x = x.toString();  typeof x;   // string    typeof 12;  // number    typeof (12).toString(); // string |

**toFixed(n)**

Hàm này có tác dụng chuyển một số sang một số có n số lẻ ở sau nó và có làm tròn.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=374)

|  |  |
| --- | --- |
| 1  2  3  4  5 | var x = 5.656;  x.toFixed(0); // returns 6  x.toFixed(2); // returns 5.66  x.toFixed(4); // returns 5.6560  x.toFixed(6); // returns 5.656000 |

**toPrecision(n)**

Hàm này có tác dụng chuyển một số thành số có chiều dài là n, hàm  này khác với hàm toFixed() ở chỗ hàm toFixed() chuyển thành số có n số lẻ ở đằng sau. Có một điều lưu ý là tham số n phải luôn luôn lớn hơn 0 và nếu bạn không truyền tham số vào thì mặc định nó lấy luôn chiều dài ban đầu.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=375)

|  |  |
| --- | --- |
| 1  2  3  4  5 | var x = 5.656;  x.toPrecision();  // returns 5.656  x.toPrecision(2); // returns 5.6  x.toPrecision(4); // returns 5.656  x.toPrecision(6); // returns 5.65600 |

**valueOf()**

Hàm valueOf() có tác dụng lấy giá trị của một biến hoặc một giá trị khác - ý nghĩa là lấy giá trị của (*cái gì đó*).

Hàm này thực chất không sử dụng nhiều vì thông thường ta lấy giá trị trực tiếp luôn.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=376)

|  |  |
| --- | --- |
| 1  2  3  4  5 | var x = 123 + 12;    x.valueOf(); // returns 135    (2 + 3).valueOf(); // returns 5 |

Ngoài kiểu number ra thì hàm valueOf()  còn có thể sử dụng được với bất kì các kiểu dữ liệu khác.

**3. Lời kết**

Vậy là mình giới thiệu xong 7 hàm thường hay sử dụng trong Javascript để xử lý Number, trong các hàm trên thì mỗi hàm có tác dụng khác nhau và tồn tại ở một mức độ khác nhau (*toàn cục, cục bộ*) nên bạn cần lưu ý kẻo sử dụng nhầm lẫn.

Một lưu ý nữa là khi bạn thực hiện các phép toán thông thường như cộng, trừ, nhân, chia thì bạn phải chắc chắn dữ liệu truyền vào phải là kiểu Number thì kết quả mới trả về chính xác, vì vậy thông thường với những dữ liệu không chắc chắn là kiểu number thì bạn nên dùng hàm parseInt() hoặc Number() để chuyển đổi trước khi tính toán.

**Bài 42: Đối tượng Date trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 19-05-2015- View: 2916

**Mục luc:**

* [1. Đối tượng Date trong Javascript](http://freetuts.net/doi-tuong-date-trong-javascript-401.html#goto-h2-0)
* [2. Định dạng (format) Date trong Javascript](http://freetuts.net/doi-tuong-date-trong-javascript-401.html#goto-h2-1)
  + [Định dạng ISO](http://freetuts.net/doi-tuong-date-trong-javascript-401.html#goto-h3-0)
  + [Định dạng Long](http://freetuts.net/doi-tuong-date-trong-javascript-401.html#goto-h3-1)
  + [Định dạng Short](http://freetuts.net/doi-tuong-date-trong-javascript-401.html#goto-h3-2)
  + [Định dạng đầy đủ](http://freetuts.net/doi-tuong-date-trong-javascript-401.html#goto-h3-3)
* [3. Lời kết](http://freetuts.net/doi-tuong-date-trong-javascript-401.html#goto-h2-2)

Trong bài này chúng ta sẽ tìm hiểu đến đối tượng Date trong Javascript, đây là đối tượng chuyên dùng để xử lý ngày giờ trên Client và đương nhiên tính ứng dụng của nó thì khỏi bàn. Bài này chúng ta chỉ đề cập một số kiến thức căn bản, còn các hàm xử lý nâng cao hơn thì chúng ta sẽ tìm hiểu ở bài tiếp theo.

**1. Đối tượng Date trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Tất cả các vấn đề liên quan đến thời gian đều nằm trong đối tượng Date, vì vậy để khởi tạo một đối tượng thời gian thì bạn dùng cú pháp sau:

|  |  |
| --- | --- |
| 1 | var timeObj = new Date(); |

Đây là cách thông thường và khi bạn dùng cách này nó sẽ lấy thời gian hiện tại trên máy của bạn (*client*), điều này có nghĩa mỗi Client sẽ có thời gian khác nhau phụ thuộc vào máy tính của họ.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=377)

|  |  |
| --- | --- |
| 1  2 | var dateObj = new Date();  document.write(dateObj); |

Và sau đây là bốn cách khởi tạo thông thường nhất.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | // Thời gian hiện tại  new Date();    // Tham số truyền vào là mili giây  new Date(milliseconds);    // Tham số truyền vào là chuỗi ngày tháng  new Date(dateString);    // Tham số truyền vào gồm  //  - year:         năm  //  - month:        tháng  //  - day:          ngày  //  - hours:        giờ  //  - minutes:      phút  //  - seconds:      giây  //  - milliseconds: mini giây  new Date(year, month, day, hours, minutes, seconds, milliseconds); |

Khi bạn truyền tham số vào thì đối tượng đó sẽ tự nhận diện và chuyển đổi về đúng định dạng ngày tháng nên bạn hoàn toàn yên tâm mà sử dụng.

**Ví dụ**: Khởi tạo một đối tượng với giá trị là ngày 20/11/2013

[XEM DEMO](http://freetuts.net/editor.html?id=378)

|  |  |
| --- | --- |
| 1 | var dateObj = new Date(2013, 11, 20); |

Cái khó ở phần này là định dạng format của ngày tháng khi in ra ngoài trình duyệt rất khó đọc, điều này cũng bình thường bởi vì nó có quy tắc format riêng.

**2. Định dạng (format) Date trong Javascript**

Có ba định dạng chính đó là:

* ISO
* Long
* Short

**Định dạng ISO**

Định dạng chuẩn của ISO 8601 là (YYYY-MM-DD) hoặc (YYYY-MM) hoặc (YYYY).

Có một lưu ý nếu bạn truyền vào không đủ (*ngày - tháng - năm - giờ - phút - giây*) thì mặc định các tham số khác sẽ lấy thời gian nhỏ nhất.

**Vi dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=379)

|  |  |
| --- | --- |
| 1  2  3 | var ISO\_1 = new Date("2014-11-20");  var ISO\_2 = new Date("2014-11");  var ISO\_3 = new Date("2014"); |

**Định dạng Long**

Định dạng Long tức là bạn truyền vào với tên của tháng là ba chữ cái đầu tiên ghi bằng tiếng Anh, lúc này bạn có thể đặt vị trí của nó thế nào cũng được vì đối tượng Date tự nhận diện và chuyển đổi.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=380)

|  |  |
| --- | --- |
| 1  2  3 | var LONG\_1 = new Date("Mar 25 2015");  var LONG\_2 = new Date("2015 Mar 25");  var LONG\_3 = new Date("25 2015 Mar"); |

Chạy lên các bạn thấy cả ba ví dụ đều có kết quả như nhau.

**Định dạng Short**

Định dạng Short được lưu trữ dưới dạng MM/DD/YYYY hoặc YYYY/MM/DD hoặc MM-DD-YYYY hoặc YYYY-MM-DD/

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=381)

|  |  |
| --- | --- |
| 1  2  3  4 | var SHORT\_1 = new Date("03-25-2015");  var SHORT\_2 = new Date("03/25/2015");  var SHORT\_3 = new Date("2015/03/25");  var SHORT\_4 = new Date("2015-03-25"); |

**Định dạng đầy đủ**

Trên là những định dạng ghi tắt, nếu truyền đầy đủ thì bạn phải truyền đẩy đủ (*ngày - tháng - năm - giờ - phút - giây - timezone*):

|  |  |
| --- | --- |
| 1 | var d = new Date("Wed Mar 25 2015 09:56:24 GMT+0100 (W. Europe Standard Time)"); |

**3. Lời kết**

Vậy để khởi tạo một đối tương thời gian thì ta dùng từ khóa new Date(), đồng thời chúng ta có bốn cách truyền tham số lúc khởi tạo. Tương ứng với mỗi cách truyền thì đối tượng Date()sẽ tự nhận diện và chuyển đổi sang đúng định dạng của nó.

Bài này mình xin dừng ở đây, bài tiếp theo chúng ta sẽ tìm hiểu các hàm xử lý ngày tháng trong Javascript và mình tin rằng bài đó sẽ cực kì hấp dẫn đấy.

**Bài 43: Hàm xử lý ngày tháng (Date) trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 19-05-2015- View: 9414

**Mục luc:**

* [1. Các hàm nhóm Date Get trong Javascript](http://freetuts.net/ham-xu-ly-ngay-thang-date-trong-javascript-400.html#goto-h2-0)
* [2. Các nhàm nhóm Date Set trong Javascript](http://freetuts.net/ham-xu-ly-ngay-thang-date-trong-javascript-400.html#goto-h2-1)
* [3. Tạo đồng hồ online bằng Javascript](http://freetuts.net/ham-xu-ly-ngay-thang-date-trong-javascript-400.html#goto-h2-2)
  + [Hàm checkTime()](http://freetuts.net/ham-xu-ly-ngay-thang-date-trong-javascript-400.html#goto-h3-0)
  + [Hàm startTime()](http://freetuts.net/ham-xu-ly-ngay-thang-date-trong-javascript-400.html#goto-h3-1)
* [4. Lời kết](http://freetuts.net/ham-xu-ly-ngay-thang-date-trong-javascript-400.html#goto-h2-3)

Bài trước chúng ta đã tìm hiểu đối tượng [Date trong Javascript](http://freetuts.net/doi-tuong-date-trong-javascript-401.html) rồi thì trong bài này mình sẽ nói đến một số hàm xử lý liên quan đến đối tượng Date này. Bài này khá quan trọng khi bạn viết các ứng dụng có liên quan đến ngày giờ, ví dụ ứng dụng hiển thị đồng hồ, hiển thị thời gian count down trong các chương trình deal giảm giá.

Chúng ta chia làm hai nhóm hàm chính đó là nhóm **Date Get** và nhóm **Date Set**.

**1. Các hàm nhóm Date Get trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Trong Javascript tổng cộng có 10 hàm thiết lập thời gian thông dụng:

* **getDate()** lấy ngày (1 - 31)
* **getDay()** lấy ngày trong tuần (0-6)
* **getFullYear()** lấy năm đầy đủ (YYYY)
* **getYear()** lấy năm 2 số cuối (YY)
* **getHours()** lấy số giờ (0 - 23)
* **getMiliSeconds()** lấy số mili giây (0 - 999)
* **getMinutes()** lấy số phút (0 - 59)
* **getMonth()**lấy tháng (0 - 11)
* **getSeconds()** lấy số giây (0 - 59)
* **getTime()** thời gian đã được convert sang dạng miliseconds.

Cách sử dụng các hàm trên khá đơn giản, vì nó là các phương thức của đối tượng Date nên bạn chỉ việc gọi ra và dùng.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=382)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | // Đối tượng thời gian hiện tại  var d = new Date();    d.getDate();  d.getDay();  d.getFullYear();  d.getYear();  d.getHours();  d.getMilliseconds();  d.getMinutes();  d.getMonth();  d.getSeconds();  d.getTime(); |

**Lưu ý**: với hàm lấy ngày trong tuần getDay() bạn phải cộng lên một thì mới có kết quả chính xác vì nó tính từ 0.

**2. Các nhàm nhóm Date Set trong Javascript**

Tương ứng với mỗi hàm **Date Get** thì sẽ có một hàm **Date Set** (*trừ hàm getDay()*).

* **setDate()** thiết lập ngày (1 - 31)
* **setFullYear()** thiết lập năm đầy đủ (YYYY)
* **setYear()** thiết lậpnăm 2 số cuối (YY)
* **setHours()** thiết lập số giờ (0 - 23)
* **setMiliSeconds()** thiết lập số mili giây (0 - 999)
* **setMinutes()** thiết lập số phút (0 - 59)
* **setMonth()**thiết lập tháng (0 - 11)
* **setSeconds()** thiết lập số giây (0 - 59)
* **setTime()** thiết lập thời gian đã được convert sang dạng miliseconds.

**Lưu ý**:

* Vì đây là hàm **set** nên bạn phải truyền tham số vào.
* Các hàm có ảnh hưởng lẫn nhau nhé các bạn, ví dụ bạn thiết lập ngày giờ không đúng thì nó sẽ lấy ngày giờ mặc định.
* Nếu bạn dùng hàm setTime() để thiết lập thì nó ảnh hưởng tới tất cả các giá trị còn lại bởi vì setTime() là hàm thiết lập **thời gian đầy đủ** đã chuyển sang dạng miniseconds.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=383)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | // Đối tượng thời gian hiện tại  var d = new Date();    d.setDate(20);  d.setFullYear(2011);  d.setHours(2);  d.setMilliseconds(2);  d.setMinutes(3);  d.setMonth(4);  d.setSeconds(5); |

**3. Tạo đồng hồ online bằng Javascript**

Để các bạn hiểu rõ hơn về ứng dụng của các hàm trong đối tượng Date thì mình sẽ làm một ví dụ hiển thị đồng hồ online, đây là đồng hồ được lấy từ giờ của hệ thống máy tính Client.

Trước tiên bạn cần [XEM DEMO](http://freetuts.net/editor.html?id=384) để dễ hình dung hơn, sau khi xem xong thì ta tiến hành làm nhé.

Bạn tạo một file index.html với nội dung như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <!DOCTYPE html>  <html>      <head>          <script>            // Hàm khởi tạo đồng hồ            function startTime()            {              }              // Hàm này có tác dụng chuyển những số bé hơn 10 thành dạng 01, 02, 03, ...            function checkTime(i)            {              }          </script>      </head>      <body onload="startTime()">            <div id="timer"></div>        </body>  </html> |

**Trong đó có**:

* Một thẻ div#result dùng để hiển thị đồng hồ
* Thẻ body có sự kiện onload="startTime()"  dùng để chạy đồng hồ khi website được load lên.
* Có hàm startTime() dùng để tạo đồng hồ và hàm checkTime() dùng chuyển đổi định dạng những con số sang dạng 01, 02, 03, ...

**Hàm checkTime()**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Hàm này có tác dụng chuyển những số bé hơn 10 thành dạng 01, 02, 03, ...  function checkTime(i)  {      if (i < 10) {          i = "0" + i;      }      return i;  } |

**Hàm startTime()**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | // Hàm khởi tạo đồng hồ  function startTime()  {      // Lấy Object ngày hiện tại      var today = new Date();        // Giờ, phút, giây hiện tại      var h = today.getHours();      var m = today.getMinutes();      var s = today.getSeconds();        // Chuyển đổi sang dạng 01, 02, 03      m = checkTime(m);      s = checkTime(s);        // Ghi ra trình duyệt      document.getElementById('timer').innerHTML = h + ":" + m + ":" + s;        // Dùng hàm setTimeout để thiết lập gọi lại 0.5 giây / lần      var t = setTimeout(function() {          startTime();      }, 500);  } |

Với cách làm này thì chỉ lấy được thời gian của hệ thống Client, còn nếu bạn muốn lấy thời gian từ Server thì phải thông qua PHP, đồng thời dùng hàm [setTimeout()](http://freetuts.net/settimeout-va-setinterval-trong-javascript-391.html) để tăng giờ, phút, giây lên chứ không phải lấy trực tiếp như thế này.

**4. Lời kết**

Ngoài các hàm trên thì vẫn còn rất nhiều hàm khác nhưng mình không liệt kê ra hết được, vì vậy bạn có thể tham khảo thêm [tại đây](http://www.w3schools.com/jsref/jsref_tojson.asp).

Để sử dụng rành các hàm này thì bạn phải làm một vài ví dụ nữa nhưng vì đây là serie viết về lý thuyết nên mình sẽ không đưa ra thêm. Tuy nhiên mình có một serie khác là [bài tập Javascript](http://freetuts.net/tag/bai-tap-javascript)nên bạn có thể theo dõi các ví dụ ở đó.

**Bài 44: Lệnh switch case trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 19-05-2015- View: 2322

**Mục luc:**

* [1. Lệnh switch case trong Javascript](http://freetuts.net/lenh-switch-case-trong-javascript-402.html#goto-h2-0)
* [2. Ví dụ lệnh switch case trong Javascript](http://freetuts.net/lenh-switch-case-trong-javascript-402.html#goto-h2-1)
  + [Trường hợp không có default](http://freetuts.net/lenh-switch-case-trong-javascript-402.html#goto-h3-0)
  + [Trường hợp không có break](http://freetuts.net/lenh-switch-case-trong-javascript-402.html#goto-h3-1)
  + [Trường hợp gom nhóm case](http://freetuts.net/lenh-switch-case-trong-javascript-402.html#goto-h3-2)
* [3. Lời kết](http://freetuts.net/lenh-switch-case-trong-javascript-402.html#goto-h2-2)

Lẽ ra bài này mình viết ngay từ đầu nhưng do việc học Javascript có sử dụng nhiều đối tượng riêng của nó nên mình giới thiệu các đối tượng đó trước. Chúng ta đã được học câu [kiểm tra điều kiện if else](http://freetuts.net/lenh-kiem-tra-dieu-kien-if-else-trong-javascript-267.html) rồi, tuy nhiên vẫn còn một cách khác có chức năng tương tự đó là câu lệnh switch case.

**1. Lệnh switch case trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Lênh switch case dùng để rẻ nhánh chương trình, ứng với mỗi nhánh sẽ có một điều kiện cụ thể và điều kiện đó ở dạng so sánh bằng.

**Cú pháp**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | switch (variable)  {      case value\_1 : {          // do some thing          break;      }      case value\_2 : {          // do some thing          break;      }      default : {          // do something      }  } |

Nếu như trong các **cases**không có  case nào phù hợp thì nó sẽ chạy lệnh ở default, đồng thời nếu có case nào phù hợp rồi thì lệnh break sẽ thoát khỏi lệnh switch luôn, còn nếu bạn không thêm lệnh break thì nó sẽ tiếp tục kiểm tra và chạy tiếp ở case tiếp theo.

**Quy trình chạy như sau**:

* Nếu tham số variable có giá trị là value\_1 thì những đoạn code nằm bên trong case 1 sẽ được thực hiện, ngược lại nó sẽ nhảy xuống case tiếp theo.
* Lúc này nếu variable có giá trị là value\_2 thì những đoạn code trong case 2 sẽ được thực hiện, ngược lại nó kiểm tra tiếp xem còn case nào không.
* Nhận thấy không còn case nào nữa nên nó sẽ kiểm tra có lệnh default không? Vì có lệnh default nên nó sẽ chạy đoạn code trong lệnh default đó rồi thoát khỏi switch case.

**Ví dụ**: Viết chương trình cho người dùng nhập vào một số, kiểm tra số đó là số chẵn hay số lẻ.

Với bài toán này thì mình kết hợp lệnh [prompt()](http://freetuts.net/ham-alert-confirm-prompt-trong-javascript-264.html) để lấy thông tin từ người dùng, đồng thời kết hợp lệnh switch case để hiển thị kết quả. Có một lưu ý bạn nên sử dụng hàm [parseInt()](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html) để chuyển dữ liệu người dùng nhập sang number.

[XEM DEMO](http://freetuts.net/editor.html?id=385)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | var number = parseInt(prompt("Nhập số cần kiểm tra"));    var mod = (number % 2);    switch (mod)  {      case 0 : {          document.write(number + " là số chẵn");          break;      }      case 1: {          document.write(number + " là số lẽ");          break;      }      default : {          document.write("Ký tự bạn nhập không phải số");      }  } |

Với ví dụ này thì bạn hoàn toàn có thể sử dụng lệnh if else để thực hiện.

[XEM DEMO](http://freetuts.net/editor.html?id=386)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | var number = parseInt(prompt("Nhập số cần kiểm tra"));    var mod = (number % 2);    if (mod == 0){      document.write(number + " là số chẵn");  }  else if (mod == 1){      document.write(number + " là số lẽ");  }  else{      document.write("Ký tự bạn nhập không phải số");  } |

**2. Ví dụ lệnh switch case trong Javascript**

**Đề bài:**Viết chương trình cho người dùng nhập một màu, kiểm tra màu đó có phải màu đỏ (*red*) hay màu vàng (*yellow*) hay không? Nếu không phải thì thông báo cho người dùng biết nhập sai màu.

Chúng ta sẽ giải bài này bằng nhiều cách khác nhau và mỗi cách bạn sẽ học được một kinh nghiệm xử lý lệnh switch case.

**Trường hợp không có default**

Trường hợp này nếu bạn nhập một màu khác với màu đỏ (*red*) và vàng (*yellow*) thì sẽ không có thông báo gì.

[XEM DEMO](http://freetuts.net/editor.html?id=387)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var color = prompt("Nhập màu cần kiểm tra");    switch (color){      case 'red' :          document.write("Bạn nhập màu đỏ, đúng rồi đó");          break;      case 'yellow' :          document.write("Bạn nhập màu vàng, đúng rồi đó");          break;  } |

**Trường hợp không có break**

Trường hợp này nếu bạn nhập vào màu đỏ (*red*) thì chương trình sẽ in ra cả lệnh ở case màu vàng (*yellow*) phía dưới, lý do là trong case màu đỏ ta không sử dụng lệnh break để thoát khỏi lệnh switch nên nó sẽ chạy thẳng xuống case phía dưới luôn mà không cần kiểm tra điều kiện.

[XEM DEMO](http://freetuts.net/editor.html?id=388)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | var color = prompt("Nhập màu cần kiểm tra");    switch (color){      case 'red' :          document.write("Bạn nhập màu đỏ, đúng rồi đó");      case 'yellow' :          document.write("Bạn nhập màu vàng, đúng rồi đó");          break;      default :          document.write("Màu bạn nhập không có trong hệ thống");  } |

Giả sử bạn thêm một case nữa cho màu xanh (*blue*), lúc nếu bạn nhập vào màu đỏ (*red*) thì kết quả chỉ in thêm lệnh màu vàng (*yellow*) thôi. Từ đó suy ra rằng nếu không có break thì nó chỉ chạy luôn case đầu tiên phía dưới nó.

[XEM DEMO](http://freetuts.net/editor.html?id=389)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | var color = prompt("Nhập màu cần kiểm tra");    switch (color){      case 'red' :          document.write("Bạn nhập màu đỏ, đúng rồi đó");      case 'yellow' :          document.write("Bạn nhập màu vàng, đúng rồi đó");          break;      case 'blue' :          document.write("Bạn nhập màu xanh, đúng rồi đó");          break;      default :          docment.write("Màu bạn nhập không có trong hệ thống");  } |

**Trường hợp gom nhóm case**

Nếu bạn để ý kỹ hơn thì thì thấy rằng nếu người dùng nhập vào màu đỏ (*red*), vàng (*yellow*) và xanh (*blue*) thì đều có thông báo nhập đúng. Vậy tai sao mình không gom ba trường hợp đó thành một thôi.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=390)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | var color = prompt("Nhập màu cần kiểm tra");    switch (color){      case 'red' :      case 'yellow' :      case 'blue' :          document.write("Bạn nhập màu " + color + ", đúng rồi đó");          break;      default :          document.write("Màu bạn nhập không có trong hệ thống");  } |

**3. Lời kết**

Từ ba trường hợp trên ta suy ra được kết luận như sau:

* Để xử lý cho một trường hợp ngoại lệ nào đó thì ta sử dụng lệnh default.
* Nếu nhiều nhánh cùng xử lý chung một đoạn code thì ta sẽ bỏ đi lệnh break ở bên trong, như vậy nó sẽ chạy luôn case ở phía dưới mà không cần kiểm tra điều kiện.

**Bài 45: Tìm hiểu JavaScript Hoisting**

Đăng bởi: TheHalfHeart- Vào ngày: 22-05-2015- View: 2439

**Mục luc:**

* [1. Hoisted trong Javascript](http://freetuts.net/tim-hieu-javascript-hoisting-403.html#goto-h2-0)
* [2. Không phải hoisted trong Javascript](http://freetuts.net/tim-hieu-javascript-hoisting-403.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/tim-hieu-javascript-hoisting-403.html#goto-h2-2)

Hoisting là vấn đề liên quan đến cách [khai báo biến trong Javascript](http://freetuts.net/bien-va-khai-bao-bien-trong-javascript-265.html). Nó liên quan đến việc trong Javascript bạn có thể sử dụng một biến mà không cần phải định nghĩa trước, vì vậy để chương trình chạy chuẩn thì bạn phải khai báo biến trước khi sử dụng và đặt nó phía trên cùng phạm vi của biến (*trong function, trong đoạn script, ...*).

**1. Hoisted trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Trong Javascript bạn có thể định nghĩa một biến sau khi sử dụng nó, hay nói cách khác một biến có thể được sử dụng trước và định nghĩa sau.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=406)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Gán nhưng chưa khai báo biến  domain = '[http://freetuts.net](http://freetuts.net/)';    // In giá trị  document.write("Domain là: " + domain);    // Khai báo  var domain;    // In lại  document.write("<br/> Domain là: " + domain); |

Nếu trong lúc khởi tạo mà bạn gán giá trị cho biến thì kết quả sẽ khác

[XEM DEMO](http://freetuts.net/editor.html?id=408)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Gán nhưng chưa khai báo biến  domain = '[http://freetuts.net](http://freetuts.net/)';    // In giá trị  document.write("Domain là: " + domain);    // Khai báo  var domain = '[http://course.freetuts.net](http://course.freetuts.net/)';    // In lại  document.write("<br/> Domain là: " + domain); |

Nhưng nếu viết cách khai báo biến trước khi sử dụng thì ta vẫn có kết quả giống nhau.

[XEM DEMO](http://freetuts.net/editor.html?id=407)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Khai báo  var domain;    // Gán nhưng chưa khai báo biến  domain = '[http://freetuts.net](http://freetuts.net/)';    // In giá trị  document.write("Domain là: " + domain);    // In lại  document.write("<br/> Domain là: " + domain); |

Và đây chính là cách chuẩn nhất nên viết và ta gọi là hoisting, tức là khai báo biến nằm trên cùng của đoạn mã script.

**2. Không phải hoisted trong Javascript**

Trong Javascript hoists chỉ tồn tại khi bạn khai báo biến chứ không tồn tại khi gán giá trị ban đầu cho biến, nghĩa là bạn có thể gán thoải mái và nó không liên quan đến việc dễ debug hay không, miễn là cứ khai báo ở trên top.

**Ví dụ**: [XEM DEMO](http://freetuts.net/editor.html?id=409)

|  |  |
| --- | --- |
| 1  2  3  4  5 | var domain = '[http://freetuts.net](http://freetuts.net/)';  var email = 'thehalfheart@gail.com';    document.write("Domain là: " + domain);  document.write("<br/> Email là: " + email); |

Trong ví dụ này ta khai báo và gán giá trị khởi tạo luôn.

**Ví dụ**:  [XEM DEMO](http://freetuts.net/editor.html?id=410)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var domain = '[http://freetuts.net](http://freetuts.net/)';    document.write("Domain là: " + domain);    document.write("<br/> Email là: " + email);    var email = 'thehalfheart@gail.com'; |

Trong ví dụ này ta sử dung rồi mới khai báo, lúc này chương trình chưa hiểu biến email là gì, vì vậy nó sẽ in ra là *undefined.*

**3. Lời kết**

Việc hiểu cơ chế hoisted giúp lập trình viên giảm lỗi và dễ debug trong quá trình phát triển. Không những vậy mà việc khai báo biến nằm trên top của chức năng giúp chương trình sáng và dễ quản lý code hơn.

Những ví dụ này khá đơn giản nên bạn hãy tự mình liên tưởng tới khái niệm [biến toàn cục và biến cục bộ](http://freetuts.net/bien-toan-cuc-va-bien-cuc-bo-trong-javascript-377.html), sau đó áp dụng vào thử nhé.

**Bài 46: Use Strict là gì? Strict Mode trong javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 22-05-2015- View: 5192

**Mục luc:**

* [1. Chế độ Strict Mode là gì ?](http://freetuts.net/use-strict-la-gi-strict-mode-trong-javascript-407.html#goto-h2-0)
* [2. Use Strict là gì ?](http://freetuts.net/use-strict-la-gi-strict-mode-trong-javascript-407.html#goto-h2-1)
  + [Phạm vi Strict Mode toàn cục](http://freetuts.net/use-strict-la-gi-strict-mode-trong-javascript-407.html#goto-h3-0)
  + [Phạm vi Strict Mode cục bộ](http://freetuts.net/use-strict-la-gi-strict-mode-trong-javascript-407.html#goto-h3-1)
* [3. Ví dụ Use Strict trong Javsacript](http://freetuts.net/use-strict-la-gi-strict-mode-trong-javascript-407.html#goto-h2-2)
* [4. Lời kết](http://freetuts.net/use-strict-la-gi-strict-mode-trong-javascript-407.html#goto-h2-3)

Khi bạn download những file JS trên mạng về hoặc download những Plugin jQuery thì bạn sẽ thấy phía trên cùng của file người ta có đặt dòng chữ "use strict";. Vậy câu hỏi được đặt ra là từ khóa **use strict là gì** và mục đích người ta đặt nó ở đầu file để làm gì?. Để trả lời câu hỏi trên thì ta phải tìm sự thoải mái trong việc phát triển của Javascript và sự ra đời của **Strict Mode**.

**1. Chế độ Strict Mode là gì ?**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Khi bạn làm việc với Javascript thì bạn có thể sử dụng biến mà chưa cần định nghĩa ([hoisting](http://freetuts.net/tim-hieu-javascript-hoisting-403.html)), bạn có thể quên đặt dấu chấm phẩy ở cuối mỗi đoạn code và bạn có thể sử dụng tên những từ khóa để tạo tên biến. Điều này làm cho các lập trình viên có thói quen làm việc quá dễ dãi và dẫn đến chương trình thiếu tính an toàn. Chính vì vậy Javascript kể từ phiên bản 1.8.5 trở đi đã bổ sung một khái niệm gọi **Strict Mode** nhằm giải quyết tính thiếu an toàn này.

Và bây giờ câu hỏi đặt ra là làm thế nào để khai báo sử dụng chế độ Strict Mode? Để trả lời câu hỏi này thì ta qua phần tiếp theo.

**2. Use Strict là gì ?**

**Use strict** là từ khóa khai báo sử dụng chế độ **Strict Mode**, nghĩa là nếu bạn muốn sử dụng chế độ Strict Mode ở đâu thì chỉ việc đặt từ khóa "use strict" ở đó. Chế độ Strict Mode có hai phạm vi sử dụng đó là toàn cục và cục bộ. Tính toàn cục tức là khi bạn đặt từ "use strict" ở ngoài hàm và nằm phía trên cùng của file thì lúc này tất cả các đoạn code bên dưới đều bị ảnh hưởng. Tính cục bộ tức là bạn đặt "use strict" nằm trong một hàm nào đó thì phạm vi ảnh hưởng chỉ nằm trong hàm đó mà thôi.

**Phạm vi Strict Mode toàn cục**

Trong ví dụ này chương trình chạy bị sẽ bị sai vì biến domain chưa được khởi tạo. Tuy nhiên nếu ta bỏ từ khóa **use strict** đi thì chương trình sẽ chạy bình thường.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Phía trên cùng của file  "use strict";    // Đoạn code này lỗi vì biến domain chưa được khởi tạo  domain = "[http://freetuts.net](http://freetuts.net/)";    // In ra màn hình  document.write(domain); |

Bây giờ ta thử gom đoạn code trên vào một hàm thử.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | // Phía trên cùng của file  "use strict";    function show\_domain(){      // Đoạn code này lỗi vì biến domain chưa được khởi tạo      domain = "[http://freetuts.net](http://freetuts.net/)";        // In ra màn hình      document.write(domain);  }    show\_domain(); |

Chạy lên chương trình vẫn bị lỗi vì ta đã khai báo tính toàn cục cho chế độ Strict Mode.

**Phạm vi Strict Mode cục bộ**

Sử dụng ví dụ trên nhưng ta sẽ khai báo ở trong phạm vi của hàm.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | function show\_domain(){      "use strict";  }    // Đoạn code này lỗi vì biến domain chưa được khởi tạo  domain = "[http://freetuts.net](http://freetuts.net/)";    // In ra màn hình  document.write(domain); |

Chạy lên chương trình vẫn hoạt động bình thường. Tuy nhiên nếu ta sử dụng sai trong hàm đó thì chương trình sẽ không hoạt động.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | function show\_domain(){      "use strict";        // Đoạn code này lỗi vì biến domain chưa được khởi tạo      domain = "[http://freetuts.net](http://freetuts.net/)";        // In ra màn hình      document.write(domain);  }    show\_domain(); |

**3. Ví dụ Use Strict trong Javsacript**

Bây giờ mình sẽ liệt kê ra một số trường hợp bị lỗi khi chạy ở chế độ **Strict Mode**.

**Sử dụng biến chưa được định nghĩa**: Bạn không thể sử dụng một biến mà chưa được định nghĩa trước đó.

|  |  |
| --- | --- |
| 1  2  3  4 | "use strict";    // sai vì biến domain chưa được khởi tạo trừ trước  domain = 'freetuts.net'; |

**Không chấp nhận delete biến**: Bạn không thể delete các hàm, biến nếu chạy ở chế độ Strict Mode

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | "use strict";    var domain = 'freetuts.net';    // Sai vì không được delete  delete domain; |

**Định thuộc tính nghĩa hai lần**: Nếu trong một Object bạn định nghĩa tên key bị trùng thì sẽ bị lỗi.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | "use strict";    // Sai vì key email bị trùng  var info = {      email : "thehalfheart@gmail.com",      email : "freetuts.net@gmail.com"  }; |

**Khao báo tham số bị trùng**: Nếu bạn khai báo các tham số bị trùng tên thì sẽ bị lỗi

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | "use strict";    // Sai vì tham số domain bị trùng  function show\_domain(domain, domain){      // do some thing  } |

**Lỗi literals và escape với number**: Bạn không được sử dụng literals và escape với kiểu number.

|  |  |
| --- | --- |
| 1  2  3  4  5 | "use strict";    // Sai  var x = 0100;  var y = \0100; |

**Khai báo tên biến trùng với key**: Bạn không thể khai báo tên của biến trùng với key của Javascript.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Lỗi: Trùng hàm eval  var eval = 12;    // Lỗi: từ khóa arguments không được sử dụng làm tên biến  var arguments = 12;    // Lỗi: Từ khóa delete là khóa nên không được sử dụng làm tên biến  var delete = 123; |

Ngoài ra bạn không thể sư dụng tên biến với các từ khóa sau:

* implements
* interface
* package
* private
* protected
* public
* static
* yield

**4. Lời kết**

Qua bài này bạn sẽ không còn thắc mắc **use strict là gì** nữa phải không nào. Và thông qua từ khóa **use strict** bạn đã biết thế nào là chế độ chạy Strict Mode và công dụng của nó trong việc lập trình. Vì vậy kể từ bây giờ bạn hãy tập làm quen với chế độ Strict Mode này nhé vì nó chặt chẽ và an toàn, đảm bảo chương trình của bạn có thể chạy trên mọi trình duyệt hiện nay.

**Bài 47: Đối tượng (object) trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 22-05-2015- View: 8541

**Mục luc:**

* [1. Đối tượng (Object) là gì ?](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h2-0)
  + [Khởi tạo đối tượng](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-0)
* [2. Thuộc tính và phương thức đối tượng](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h2-1)
  + [Thuộc tính](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-1)
  + [Phương thức](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-2)
  + [Xem danh sách phương thức và thuộc tính](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-3)
* [3. Thao tác với thuộc tính và phương thức đối tượng](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h2-2)
  + [Gán giá trị cho thuộc tính](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-4)
  + [Lấy giá trị của thuộc tính](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-5)
  + [Gọi phương thức](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-6)
* [4. Mảng chứa đôi tượng - đối tượng chứa đối tượng](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h2-3)
  + [Đối tượng chứa đối tượng](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-7)
  + [Mảng chứa đối tượng](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h3-8)
* [5. Lời kết](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h2-4)
* [# Câu hỏi và bài tập](http://freetuts.net/doi-tuong-object-trong-javascript-408.html#goto-h2-5)

Như ta biết Javascript không được xếp vào danh sách những ngôn ngữ [lập trình hướng đối tượng](http://freetuts.net/tag/lap-trinh-huong-doi-tuong), tuy nhiên về khách quan mà nói thì bạn hoàn toàn có thể lập trình hướng đối tượng thông qua kiểu dữ liệu Object mà Javascript đã cung cấp. Chi tiết thế nào thì chúng ta cùng tìm hiểu nhé.

**1. Đối tượng (Object) là gì ?**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Trong Javascript đối tượng là một khái niệm trừu tượng thể hiện cho một đối tượng cụ thể và nó có sẵn một số đối tượng như [Date](http://freetuts.net/doi-tuong-date-trong-javascript-401.html), [Number](http://freetuts.net/doi-tuong-number-trong-javascript-397.html). Ngoài các đối tượng này thì lập trình viên có thể tự tạo một đối tượng theo ý của mình dựa vào yêu cầu của ứng dụng. Ví dụ mình cần tạo ra một đối tượng chuyên xử lý vấn đề về bình luận cho trang tin tức thì mình sẽ tạo một đối tượng Comment.

**Khởi tạo đối tượng**

Đấy là về mặt lý thuyết, còn về mặt thực hành thì để tạo một đối tượng bạn sẽ có hai cách sau.

**Cách 1**: Sử dụng từ khóa new Object()

|  |  |
| --- | --- |
| 1 | var Comment = new Object(); |

**Cách 2**: Sử dụng từ khóa {}

|  |  |
| --- | --- |
| 1 | var Comment = {}; |

**2. Thuộc tính và phương thức đối tượng**

Mỗi đối tượng sẽ có các thuộc tính và phương thức.

**Thuộc tính**

Thuộc tính là những đặc điểm (*có thể hiểu là biến*) cần lưu trữ trong đối tượng. Ví dụ với đối tượng Comment thì mình cần các thuộc tính sau.

* title
* content
* fullname
* email

Lúc này ta có thể khai báo bằng ba cách.

**Cách 1**: Sử dụng từ khóa new Object()

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Khởi tạo  var Comment = new Object();    // Thêm thuộc tính  Comment.title = '';  Comment.content = '';  Comment.fullname = '';  Comment.email = ''; |

**Cách 2**: Sử dụng từ khóa {} và thêm thuộc tính ngay lúc khai báo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // Khởi tạo  var Comment = {      title : "",      content : "",      fullname : "",      email : ""  }; |

**Cách 3**: Sử dụng từ khóa {} và thêm thuộc tính sau đó

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Khởi tạo  var Comment = {};    // Thêm thuộc tính  Comment.title = '';  Comment.content = '';  Comment.fullname = '';  Comment.email = ''; |

Trong ba cách trên thì mình khuyến khích các bạn nên sử dụng cách thứ hai bởi vì nó mạch lạc và dễ quản lý code hơn.

**Phương thức**

Phương thức là những hành động (*có thể hiểu là hàm*) của đối tượng. Ví dụ trong đối tượng Comment thì mình cần hai phương thức là:

* addComment()
* validateComment()

Lúc này ta sẽ có ba cách khai báo tương tự như cách khai báo thuộc tính.

**Cách 1**: Sử dụng từ khóa new Object()

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Khởi tạo  var Comment = new Object();    // Thêm phương thức  Comment.addComment = function(){      // do some thing  };    Comment.validateComment = function(){      // do some thing  }; |

**Cách 2**: Sử dụng từ khóa {} và thêm phương thức ngay lúc khai báo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // Khởi tạo  var Comment = {      addComment : function(){          // do some thing      },      validateComment : function(){          // do some thing      }  }; |

**Cách 3**: Sử dụng từ khóa {} và thêm phương thức sau đó

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Khởi tạo  var Comment = {};    // Thêm phương thức  Comment.addComment = function(){      // do some thing  };    Comment.validateComment = function(){      // do some thing  }; |

Và mình cũng khuyến khích các bạn nên sử dụng cách 2.

**Xem danh sách phương thức và thuộc tính**

Cách nhanh nhất để xem và debug đối tượng thì bạn nên sử dụng [Firebug](http://freetuts.net/su-dung-firebug-de-kiem-tra-loi-javascript-380.html). Ví dụ đối tượng Comment trên thì khi sử dụng Firebug kết hợp hàm [console.log()](http://freetuts.net/lenh-consolelog-ket-hop-voi-firebug-trong-javascript-381.html) thì sẽ có kết quả như hình sau:

http://freetuts.net/upload/tut_post/images/2015/05/22/408/doi-tuong-object-trong-javascript.png

**3. Thao tác với thuộc tính và phương thức đối tượng**

Sau khi tạo xong đối tượng thì ta có hai cách sử dụng căn bản đó là gọi và gán dữ liệu cho thuộc tính và phương thức. Nhưng nếu xem xét mọi khía cạnh thì chúng ta có các thao tác thông thường như sau:

* Gán giá trị cho thuộc tính
* Lấy giá trị của thuộc tính
* Gọi phương thức

**Gán giá trị cho thuộc tính**

Để gán giá trị cho thuộc tính chúng ta chỉ việc thực hiện bằng cách sử dụng toán tử =giống như cách gán giá trị cho biến.

|  |  |
| --- | --- |
| 1 | Comment.title = "Tiêu đề bình luận"; |

Nhưng nếu bạn gọi từ một hàm trong đối tượng thì bạn có thể sử dụng từ khóa this.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var Comment = {      title : "",      addComment : function(){          this.title = "Tiêu đề bình luận";      }  }; |

**Lấy giá trị của thuộc tính**

Để lấy giá trị thuộc tính thì ta làm tương tự như thao tác với biến.

|  |  |
| --- | --- |
| 1 | var title = Comment.title; |

Nếu gọi từ một hàm trong đối tượng thì bạn có thể sử dụng từ khóa this.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var Comment = {      title : "",      addComment : function(){          var title = this.title;      }  }; |

**Gọi phương thức**

Tương tự như thuộc tính chúng ta gọi bình thường.

|  |  |
| --- | --- |
| 1 | comment.addComment(); |

Gọi trong hàm của đối tượng.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | var Comment = {      title : "",      addComment : function(){          this.validateComment();      },      validateComment : function(){          // do some thing      }  }; |

**4. Mảng chứa đôi tượng - đối tượng chứa đối tượng**

Mỗi đối tượng (object) trong Javascript có thể chứa các đối tượng khác hoặc một mảng có thể chứa các đối tượng.

**Đối tượng chứa đối tượng**

**Ví dụ 1**: Gom các thuộc tính của Comment vào một đối tượng Info.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var Comment = {      info : {          title : "",          content : "",          email : "",          fullname : ""      }  }; |

**Ví dụ 2**: Gom các phương thức (hàm) của Comment vào đối tượng func.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var Comment = {      func : {          addComment : function(){              // Something          },          validateComment : function(){              // Something          }      },  }; |

Lúc này để truy xuất tới các thuộc tính và phương thức này ta chỉ việc sử dụng dấu chấm và bổ sung thêm một cấp nữa.

|  |  |
| --- | --- |
| 1  2 | Comment.info.title = "Comment tại freetuts.net";  Comment.func.addComment(); |

**Mảng chứa đối tượng**

Để gán giá trị là một đối tượng vào mảng cũng tương tự như gán các giá trị bình thường khác (xem [mảng trong Javascript](http://freetuts.net/thao-tac-voi-mang-trong-javascript-288.html)).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | // Đối tượng Comment  var Comment = {      title   : "",      content : "",      email   : "",      fullname : ""  };    // Khởi tạo mảng  var Comments = [];    // Gán giá trị cho phần tử mảng  Comments[0] = Comment;    // Gọi tới thuộc tính  Comments[0].title = "Tiêu đề bình luận";  alert(Comments[0].title); |

Hoặc

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Khởi tạo mảng  var Comment = [{      title   : "",      content : "",      email   : "",      fullname : ""  }];    // Sử dụng  Comment[0].title = "Tiêu đề bình luận";  alert(Comment[0].title); |

**5. Lời kết**

Nói về đối tượng trong Javascript thì rất là nhiều nên mình không thể trình bày trong một bài được, vì vậy bài này mình xin dừng ở đây và ở các bài tiếp theo chúng ta sẽ thực hàm các ví dụ liên quan đến đối tượng tìm hiểu mối liên hệ giữa đối tượng và JSON trong Javascript.

Trong các phần trên thì ở phần 5 là hơi khó hiểu nhất vì nó liên quan tới mảng và có cấu trúc rườm rà. Nhưng bạn đừng lo lắng bởi vì các bài tiếp theo mình sẽ đề cập tới nó cũng khá nhiều.

**Bài 48: Thao tác với đối tượng (Object) trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 23-05-2015- View: 4656

**Mục luc:**

* [1. Thao tác với đối tượng (Object) trong Javascript](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html#goto-h2-0)
  + [Hàm xem đối tượng sinh viên](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html#goto-h3-0)
  + [Hàm thêm đối tượng sinh viên](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html#goto-h3-1)
  + [Hàm xóa đối tượng sinh viên](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html#goto-h3-2)
  + [Hàm sửa đối tượng sinh viên](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html#goto-h3-3)
  + [Toàn bộ code cho đối tượng sinh viên](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html#goto-h3-4)
  + [Sử dụng đối tượng sinh viên](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html#goto-h3-5)
* [2. Lời kết](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html#goto-h2-1)

Ở bài trước mình đã giới thiệu qua về [Object trong Javascript](http://freetuts.net/doi-tuong-object-trong-javascript-408.html) và có hướng dẫn các thao tác như tạo mới đối tuọng, lấy giá trị và gán giá trị đối tượng. Vậy trong bài này mình tiếp tục nói về đối tượng (*Object*) và chủ đề là các thao tác với đối tượng (Object) trong Javascript.

Và để rõ ràng hơn thì mình sẽ trình bày một ví dụ về xây dựng đối tượng SinhViên, thông qua đối tượng này sẽ thể hiện tất cả  các thao tác cần thiết trong một đối tượng. Và để dễ viết bài hơn thì mình sẽ gọi đối tượng là Object nhé các bạn.

**1. Thao tác với đối tượng (Object) trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

**Đề bài**: Viết chương trình quản lý sinh viên gồm các thao tác chính như sau:

* Xem danh sách sinh viên
* Thêm sinh viên
* Xóa sinh viên khỏi danh sách
* Sửa thông tin sinh viên

Với mỗi sinh viên cần lưu trữ các thông tin sau:

* Mã sinh viên
* Tên sinh viên
* Email

**Hướng dẫn**: Trước tiên ta cần xác định danh sách các thuộc tính của đối tượng và đó chính là các thông tin lưu trữ của sinh viên. Ta cần lưu trữ danh sách sinh viên, vì vậy ta cũng cần có thêm một mảng lưu trữ danh sách sinh viên. Từ đó có thể xác định cấu trúc của đối tượng như sau.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | var Student = {      data : [],      viewStudent : function(){          // Xem danh sách sinh viên      },      addStudent : function(id, name, email){          // Thêm sinh viên mới      },      removeStudent : function(id){          // Xóa sinh viên      },      editStudent : function(id, name, email){          // Sửa sinh viên      }  }; |

Nhiệm vụ bây giờ là viết chi tiết vào các hàm cụ thể.

**Hàm xem đối tượng sinh viên**

Hàm này khá đơn giản vì ta chỉ lặp qua đối tượng và dùng hàm document.write để hiển thị thông tin.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | viewStudent : function(){      // Lấy danh sách sinh viên      var listStudent = this.data;        // Lặp và hiển thị sinh viên      for(var i = 0; i < listStudent; i++){          document.write("<div>" + listStudent[i].id + "|" + listStudent[i].name + "|" + listStudent[i].email + "</div>");      }  } |

**Hàm thêm đối tượng sinh viên**

Để thêm một đối tượng chúng ta cần truyền vào ba tham số như yêu cầu đề bài.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | addStudent : function(id, name, email){      // Tạo thông tin sinh viên      var item = {          id : id,          name : name,          email : email      };        //Thêm sinh viên      this.data.push(item);  } |

**Hàm xóa đối tượng sinh viên**

Để xóa sinh viên chúng ta cần biết id của sinh viên đó, đồng thời sử dụng hàm splice để xóa phần tử của mảng, nếu chưa biết thì bạn có thể xem bài các hàm [xử lý mảng trong Javascript](http://freetuts.net/cac-ham-xu-ly-mang-trong-javascript-289.html).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | removeStudent : function(id){      // Lặp qua sinh viên để tìm kiếm và xóa      for(var i = 0; i < this.data.length; i++){          if (this.data[i].id === id) { // nếu là sinh viên cần xóa              this.data.splice(i, 1); // thì xóa          }      }  }, |

**Hàm sửa đối tượng sinh viên**

Việc sửa sinh viên cũng  tương tự như việc thêm sinh viên, nghĩa là chúng ta cần truyền vào ba tham số như yêu cầu đề bài, sau đó dựa vào id truyền vào để tìm sinh viên cần sửa.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | editStudent : function(id, name, email){      // Tìm sinh viên cần edit      for(var i = 0; i < this.data.length; i++){          // nếu là sinh viên cần edit thì thực hiện edit          if (this.data[i].id === id) {              this.data[i].name = name;              this.data[i].email = email;          }      }  } |

Như vậy việc xử lý các phương thức trong đối tượng phụ thuộc vào từng bài toán cụ thể và có một điều chắc chắn rằng chúng có liên quan tới nhau. Và cuối cùng đây là toàn bộ đối tượng của chúng ta.

**Toàn bộ code cho đối tượng sinh viên**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | var Student = {      data : [],      viewStudent : function(){          // Lấy danh sách sinh viên          var listStudent = this.data;            // Lặp và hiển thị sinh viên          for(var i = 0; i < listStudent.length; i++){              document.write("<div>" + listStudent[i].id + "|" + listStudent[i].name + "|" + listStudent[i].email + "</div>");          }      },      addStudent : function(id, name, email){          // Tạo thông tin sinh viên          var item = {              id : id,              name : name,              email : email          };            //Thêm sinh viên          this.data.push(item);      },      removeStudent : function(id){          // Lặp qua sinh viên để tìm kiếm và xóa          for(var i = 0; i < this.data.length; i++){              if (this.data[i].id === id) { // nếu là sinh viên cần xóa                  this.data.splice(i, 1); // thì xóa              }          }      },      editStudent : function(id, name, email){          // Tìm sinh viên cần edit          for(var i = 0; i < this.data.length; i++){              // nếu là sinh viên cần edit thì thực hiện edit              if (this.data[i].id === id) {                  this.data[i].name = name;                  this.data[i].email = email;              }          }      }  }; |

**Sử dụng đối tượng sinh viên**

Chúng ta thực hiện một số thao tác để bạn nắm rõ hơn nhé.

[XEM DEMO](http://freetuts.net/editor.html?id=411)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | document.write('<h4>Danh sách sinh viên ban đầu</h4>');  Student.viewStudent();    document.write('<h4>Danh sách sinh viên sau khi thêm hai sinh viên</h4>');  Student.addStudent("sv001", 'Nguyễn Văn Cường', "thehalfheart@gmail.com");  Student.addStudent("sv002", 'Vũ Thị Thu Tình', "freetuts.net@gmail.com");  Student.viewStudent();    document.write('<h4>Danh sách sinh viên sau khi xóa một sinh viên</h4>');  Student.removeStudent('sv001');  Student.viewStudent();    document.write('<h4>Danh sách sinh viên sau khi sửa thông tin</h4>');  Student.editStudent('sv002', "Tên Sinh Viên Mới", "mrcuong.winter@gmail.com");  Student.viewStudent(); |

**2. Lời kết**

Vậy là ta đã thực hành xong kỹ năng **thao tác với Object trong Javascript**. Bản thân mình thấy bài này khá hay và có độ khó tương đối cao, hay nói cách khác đây là kiến thức Javascript nâng cao rồi nên sẽ hơi khó hiểu nếu bạn chưa nắm vững các bài trước.

Ở phần sử dụng đối tượng ban đầu mình tính xây dựng một chương trình kết hợp với HTML luôn nhưng nghĩ lại sẽ tốn thời gian nên mình chỉ đưa ra cách sử dụng rất là đơn giản, vì vậy rất mong các bạn hiểu bài.

**Bài 49: Javascript Object Prototypes**

Đăng bởi: TheHalfHeart- Vào ngày: 23-05-2015- View: 6142

**Mục luc:**

* [1. Prototypes trong Javascript](http://freetuts.net/javascript-object-prototypes-410.html#goto-h2-0)
  + [Bổ sung dữ liệu vào đối tượng](http://freetuts.net/javascript-object-prototypes-410.html#goto-h3-0)
* [2. Tạo mới Object và thao tác với Prototype](http://freetuts.net/javascript-object-prototypes-410.html#goto-h2-1)
* [3. Thêm thuộc tính và phương thức vào Prototype](http://freetuts.net/javascript-object-prototypes-410.html#goto-h2-2)
* [4. Lời kết](http://freetuts.net/javascript-object-prototypes-410.html#goto-h2-3)

Chúng ta đã tìm hiểu về [Object](http://freetuts.net/doi-tuong-object-trong-javascript-408.html) và các [thao tác với Object](http://freetuts.net/thao-tac-voi-doi-tuong-object-trong-javascript-409.html) trong Javascript rồi nhưng đó cũng chỉ là một phần khá nhỏ liên quan đến đối tượng. Lý do nó chưa thể hiện được các tính chất căn bản của một đối tượng như tạo mới đối tượng, giá trị khởi tạo cho đối tượng, ... Vậy trong bài này mình sẽ trình bày về những khuyết điểm này và đó chính là **Javascript Object Prototypes**.

**1. Prototypes trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Prototypes dịch ra là các nguyên mẫu nhưng về phía lập trình Javascript thì bạn có thể hiểu nó giống như các thành phần phương thức và  thuộc tính của một đối tượng, vì vậy tất cả các đối tượng trong Javascript đều có một Prototype riêng để lưu trữ các thành phần đó. Lúc này khi bạn tạo mới một đối tượng thì đối tượng đó sẽ kế thừa tất cả các phương thức và thuộc tính chứa trong Prototype của đối tượng đó.

Ví dụ khi bạn tạo mới một đối tượng [Number](http://freetuts.net/doi-tuong-number-trong-javascript-397.html) thì nó sẽ có tất cả các [hàm](http://freetuts.net/ham-xu-ly-number-trong-javascript-399.html) (*phương thức*) xử lý riêng của đối tượng Number.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Tạo mới đối tượng Number  var age = new Number(12);    // Lúc này sẽ sử dụng được các phương thức  age.toString();  age.toFixed();  age.toPrecision();  age.valueOf(); |

**Bổ sung dữ liệu vào đối tượng**

Vậy câu hỏi đặt ra là có thể thêm các thuộc tính và phương thức vào Prototype của một đối tượng nào đó hay không? Câu trả lời là bạn hoàn toàn có thể làm được bằng cú pháp sau:

|  |  |
| --- | --- |
| 1  2  3  4 | Object.prototype.thuoc\_tinh = "giá trị mặc định";  Object.prototype.phuong\_thuc = function(){      // some thing  }; |

**Ví dụ**: Thêm một hàm plus() cồng dồn vào giá trị hiện tại của đối tượng Number. Lưu ý bạn cần sử dụng hàm valueOf() để lấy giá trị hiện tại của đối tượng.

[XEM DEMO](http://freetuts.net/editor.html?id=412)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | // Tạo đối tượng  Number.prototype.plus = function(value){  return this.valueOf() + parseInt(value);  };    // Tạo mới đối tượng  var age = Number(12);  document.write(age.plus(12) + "<br/>");    // Tạo đối tượng khác  var year = 2014;  document.write(year.plus(12)); |

Trong ví dụ này bạn thấy mình sử dụng hai cách để tạo đối tượng Number và cả hai đều sử dụng được hàm plus().

**2. Tạo mới Object và thao tác với Prototype**

Chúng ta đã được học cách tạo mới Object với từ khóa new Object và {} rồi nhưng khi tạo bằng hai cách này ta không thể sử dụng từ khóa new để khởi tạo mới một đối tượng được.

|  |  |
| --- | --- |
| 1  2  3  4 | var Person = {};    // Sai, không hoạt động  var p = new Person(); |

Vậy có một cách khác có thể giải quyết được đó là thông qua [hàm trong Javascript](http://freetuts.net/ham-va-tao-ham-function-trong-javascript-274.html).

**Ví dụ**: Tạo đối tượng Person gồm các thuộc tính (name, email, address) và phương thức showInfo() để hiển thị thông tin.

[XEM DEMO](http://freetuts.net/editor.html?id=412)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | function Person(){        // Thuộc tính      this.name = "";      this.email = "";      this.address = "";        // Phương thức      this.showInfo = function(){          documenet.write("Tên là: " + this.name + "<br/>");          documenet.write("Email là: " + this.email + "<br/>");          documenet.write("Địa chỉ là: " + this.address + "<br/>");      };  } |

Lúc này muốn tạo mới đối tượng và sử dụng các thuộc tính, phương thức thì ta làm như sau.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | // Tạo mới  var cuong = new Person();    // Gán thuộc tính  cuong.name = "Nguyễn Văn Cường";  cuong.email = "thehalfheart@gmail.com";  cuong.address = "Buôn Ma Thuột ĐăkLăk";    // Gọi phương thức  cuong.showInfo(); |

**3. Thêm thuộc tính và phương thức vào Prototype**

Để bổ sung thuộc tính hoặc phương thức vào một đối tượng thông qua Prototype thì bạn sử dụng cú pháp ở phần 1 mà mình đã giới thiệu.

**Ví dụ**: Bổ sung thuộc tính gender và phương thức showGender vào đối tượng Person.

[XEM DEMO](http://freetuts.net/editor.html?id=413)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | // Tạo đối tượng  function Person(){        // Thuộc tính      // ...        // Phương thức      // ...  }    // Bổ sung thông tin  Person.prototype.gender = "";  Person.prototype.showGender = function(){    document.write(this.gender);  };      // Sử dụng  var cuong = new Person();  cuong.gender = "Nam";  cuong.showGender(); |

**4. Lời kết**

Prototype có vai trò rất quan trọng trong việc quản lý các phương thức và thuộc tính của đối tượng, chính vì vậy thông thường những bạn chỉ code ở mức độ đơn giản thì rất khi khi sử dụng, còn những bạn muốn code nâng cao thì chắc chắn phải biết đến đối tượng Prototype này.

Trong bài mình có đưa ra một số ví dụ rất trực quan và thể hiện được các thao tác liên quan đến Prototype trong Javascript, hy vọng qua bài này bạn sẽ hiểu và thấy được sức mạnh thực sự của Javascript.

**Bài 50: RegExp - Regular Expression trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 22-06-2015- View: 5774

**Mục luc:**

* [1. Regular Expression trong Javascript](http://freetuts.net/regexp-regular-expression-trong-javascript-405.html#goto-h2-0)
* [2. Các quy tắc Regular Expression căn bản](http://freetuts.net/regexp-regular-expression-trong-javascript-405.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/regexp-regular-expression-trong-javascript-405.html#goto-h2-2)

Bài này chúng ta sẽ tìm hiểu đến chuỗi và cách sử dụng biểu thức chính quy (*Regular Expression*) để xử lý chuỗi nâng cao. Như ta biết Regular Expression hoạt động theo quy tắc so khớp va lấy kết quả của sự so khớp đó thông qua những quy tắc mà nó đưa ra. Dựa vào những quy tắc này lập trình viên sẽ viết thành những biểu thức và áp dụng vào bài toán của mình.

**Regular Expression** được ghi tắt là **RegExp**nên để ngắn gọn thì trong bài này mình sẽ nói là RegExp nhé các bạn.

**1. Regular Expression trong Javascript**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Để hiểu rõ hơn về Regular Expression thì bạn nên học qua serie tìm hiểu [Regular Expression trong PHP](http://freetuts.net/tag/regular-expression) vì trong serie đó mình viết khá chi tiết và đầy đủ. Và để không mất thời gian viết lại những gì đã viết thì phần này mình chỉ nói sơ sơ qua thôi nhé.

**Note**: Trong Javascript thì Regular Expression là một chuỗi nhưng nó không được bao quanh bởi cặp dấu nháy đơn ' hoặc nháy kép " mà nó được bao quanh bởi cặp dấu /.

**Cú pháp**:

|  |  |
| --- | --- |
| 1 | /pattern/modifiers |

**Trong đó**:

* pattern là chuỗi Regular Expression
* modifiers là thông số cấu hình cho chuỗi pattern và nó có các giá trị:
  + i : so khớp không quan tâm đến chữ hoa chữ thường
  + g : so khợp toàn bộ chuỗi cần tìm
  + m : so khớp luôn cả các dữ liệu xuống dòng (multiline)

**Ví dụ**: Pattern kiểm tra chuỗi có tồn tại chữ "*freetuts*" không, không phân biệt chữ hoa chữ thường và tìm toàn bộ tài liệu.

|  |  |
| --- | --- |
| 1 | var pattern = /freetuts/igm; |

Trong ví dụ này thì:

* pattern là freetuts
* modifiers là igm

**2. Các quy tắc Regular Expression căn bản**

Sau đây là một số quy tắc căn bản thường hay sử dụng trong Regular Expression.

| **Expression** | **Description** |
| --- | --- |
| [abc] | Tìm các chữ cái a,b hoặc c |
| [0-9] hoăc [a-z] | Tìm các ký tự từ 0-9 hoặc từ a-z |
| (x|y) | ìm ký tự x hoặc y |
| \d | Tìm các chữ số |
| \s | Tìm khoảng trắng |
| n+ | Tìm 1 hoặc nhiều chữ n liên tiếp nhau |
| n\* | Tìm 0 hoặc nhiều chữ n liên tiếp nhau |
| n? | Tìm 0 hoặc 1 chữ n |

Bây giờ mình sẽ thực hành một vài ví dụ nhé. Nhưng trước khi đi vào ví dụ thì mình xin giới thiệu một hàm trong parttern dùng để kiểm tra một chuỗi đó là hàm test().

**Cú pháp**: pattern.test(string)

**Trong đó**:

* pattern là chuỗi pattern
* string là chuỗi cần kiểm tra.

**Kết quả trả về**: TRUE nếu khớp và FALSE nếu không khớp.

**Ví dụ**: Kiểm tra trong chuỗi có xuất hiện chữ freetuts hay không?

Ví dụ này ta chỉ cần truyền nguyên chuỗi freetuts vào pattern là xử lý được.

[XEM DEMO](http://freetuts.net/editor.html?id=414)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var patt = /freetuts/;  if (patt.test("Chào mừng bạn đến với freetuts.net")) {      document.write('Trong chuỗi có chữ freetuts');  }  else {      document.write('Trong chuỗi không có chữ freetuts');  } |

Với ví dụ này bạn có thể sử dụng trực tiếp hàm test() trong pattern cũng được.

|  |  |
| --- | --- |
| 1 | /freetuts/.test("Chào mừng bạn đến với freetuts.net") |

**Ví dụ**: Kiểm tra chuỗi có ít nhất một chữ n

Ví dụ này hơi đặc biệt nên ta sử dụng dấu + trong bảng RegEx căn bản trên, nó có tác dụng kiểm tra một chuỗi có ít nhất 1 ký tự cần tìm kiếm.

[XEM DEMO](http://freetuts.net/editor.html?id=415)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/n+/.test("hello")) {      document.write('Trong chuỗi có chữ n');  }  else {      document.write('Trong chuỗi không có chữ n');  } |

**Ví dụ**: Kiểm tra trong chuỗi có xuất hiện số hay không?

Với ví dụ này ta có hai cách giải, cách thứ nhất sử dụng cặp dấu ngoặc [0-9] và cách thứ hai là dùng ký hiệu \d.

Trong ví dụ này mình giải bằng cặp dấu ngoặc [0-9].

[XEM DEMO](http://freetuts.net/editor.html?id=416)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/[0-9]/.test("hello123")) {      document.write('Trong chuỗi có xuất hiện số');  }  else {      document.write('Trong chuỗi không xuất hiện số');  } |

**Ví dụ**: Kiểm tra trong chuỗi không hoặc có xuất hiện số

Bài này ta sẽ kết hợp cặp dấu ngoặc [0-9]và ký hiệu \* trong bảng trên. Rõ ràng trường hợp này thì chuỗi RegEx luôn đúng bởi vì ta có 2 trường hợp XUẤT HIỆN hoặc KHÔNG XUẤT HIỆN.

[XEM DEMO](http://freetuts.net/editor.html?id=417)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/[0-9]\*/.test("freetuts")) {      document.write('Luôn luôn chạy');  }  else {      document.write('Không bao giờ chạy');  } |

**Ví dụ**: Kiểm tra trong chuỗi có chữ H hay không?

Bài này khá đơn giản là ta chỉ cần truyền chữ H vào như sau:

[XEM DEMO](http://freetuts.net/editor.html?id=418)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/H/.test("hello")) {      document.write('Có chữ H');  }  else {      document.write('Không có chữ H');  } |

Rõ ràng trong ví dụ có xuất hiện chữ h mà tại sao nó báo không xuất hiện? Lý do là trong chuỗi pattern chúng ta kiểm tra chữ in HOA nhưng trong chuỗi cần kiểm tra thì lại có chữ in thường nên nó báo sai. Để giải quyết thì ta thêm chữ i đằng sau chuỗi pattern.

[XEM DEMO](http://freetuts.net/editor.html?id=419)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | if (/H/i.test("hello")) {      document.write('Có chữ H');  }  else {      document.write('Không có chữ H');  } |

Các bạn xem phần **modifiers** ở trên để biết thêm thông tin.

**3. Lời kết**

Trong bài mình cũng chỉ đưa ra các ví dụ không có tính ứng dụng trong thực tế nhưng bài này giúp bạn làm quen với biểu thức chính quy **RegEx trong Javascript**. Nó thuộc dạng kiến thức nâng cao nên rất khó, vì vậy bạn hãy thực hành kỹ để bài tiếp theo dễ hiểu bài hơn. .

Còn khá nhiều bài tập nhưng bạn cần phải biết nhiều hơn nữa các biểu thức Regular Expression và vấn đề này mình sẽ trình bày ở bài tiếp theo.

**Bài 51: Bảng biểu thức Regular Expression trong Javascript**

Đăng bởi: TheHalfHeart- Vào ngày: 22-06-2015- View: 6469

**Mục luc:**

* [1. Bảng quy tắc Regular Expression](http://freetuts.net/bang-bieu-thuc-regular-expression-trong-javascript-418.html#goto-h2-0)
  + [Modifiers](http://freetuts.net/bang-bieu-thuc-regular-expression-trong-javascript-418.html#goto-h3-0)
  + [Brackets](http://freetuts.net/bang-bieu-thuc-regular-expression-trong-javascript-418.html#goto-h3-1)
  + [Metacharacters](http://freetuts.net/bang-bieu-thuc-regular-expression-trong-javascript-418.html#goto-h3-2)
  + [Quantifiers](http://freetuts.net/bang-bieu-thuc-regular-expression-trong-javascript-418.html#goto-h3-3)
* [2. Các ví dụ thực hành với biểu thức RegExp](http://freetuts.net/bang-bieu-thuc-regular-expression-trong-javascript-418.html#goto-h2-1)
* [3. Lời kết](http://freetuts.net/bang-bieu-thuc-regular-expression-trong-javascript-418.html#goto-h2-2)

Chúng ta đã được học [Regular Expression trong Javascript](http://freetuts.net/regexp-regular-expression-trong-javascript-405.html) và cũng đã làm một số bài tập kết hợp với một số quy tắc căn bản mà mình đã liệt kê. Nhưng đó chỉ là nhũng quy tắc căn bản nhất và hoàn toàn không đầy đủ. Vì vậy trong bài này mình sẽ bổ sung các biểu thức Regular Expression nâng cao.

**1. Bảng quy tắc Regular Expression**

Bài viết được đăng tại freetuts.net - không copy dưới mọi hình thức.

Chúng ta sẽ chia thành từng nhóm để sau này dễ tìm kiếm.

**Modifiers**

| **Modifier** | **Description** |
| --- | --- |
| i | So sánh không phân biệt chữ hoa chữ thường (case-insensitive) |
| g | So sánh toàn bộ chuỗi dù trong chuỗi có xuống hàng (global) |
| m | So sánh nhiều dòng (multiline) |

**Brackets**

| **Expression** | **Description** |
| --- | --- |
| [abc] | Tìm các ký tự a, b hoặc c |
| [^abc] | Tìm các ký tự không phải a, b và c |
| [0-9] | Tìm các ký tự là chữ số từ 0-9 |
| [^0-9] | Tìm các ký tự không phải chữ số từ 0-9 |
| (x|y) | Tìm ký tự x hoặc y |

**Metacharacters**

| **Expression** | **Description** |
| --- | --- |
| . | Tìm ký tự bất kì |
| \w | Tìm ký tự chữ cái |
| \W | Tìm các ký tự không phải là chữ cái |
| \d | Tìm ký tự là chữ số |
| \D | Tìm ký tự không phải là chữ số |
| \s | Tìm ký tự là khoảng trắng |
| \S | Tìm ký tự không phải khoảng trắng |
| \b | Tìm so khớp bắt đầu hoặc kết thúc chuỗi |
| \B | Tìm so khớp không phải bắt đầu hoặc kết thúc chuỗi |
| \0 | Tìm ký tự NULL |
| \n | Tìm ký tự xuống hàng |
| \t | Tìm ký tự tab |

**Quantifiers**

| **Expression** | **Description** |
| --- | --- |
| + | Kiểm tra ký tự xuất hiện một hoặc nhiều lần |
| \* | Kiểm tra ký tự xuất hiện không hoặc nhiều lần |
| ? | Kiểm tra ký tự xuất hiện không hoặc một lần |
| {X} | Kiểm tra ký tự xuất hiện đúng X lần |
| {X,Y} | Kiểm tra ký tự xuất hiện tối thiểu X lần và tối đa Y lần |
| {X,} | Kiểm tra ký tự xuất hiện ít nhất X lần |
| ^ | Kiểm tra ký tự bắt đầu chuỗi |
| $ | Kiểm tra ký tự kết thúc chuỗi |

**2. Các ví dụ thực hành với biểu thức RegExp**

Sau đây là các ví dụ thực hành để học **Regular Expression trong Javascript**.

**Bài 1**: Kiểm tra chuỗi xuất hiện ít nhất 10 chữ N liên tiếp và không phân biệt chữ hoa chữ thường

Ta sử dụng cú pháp {10,} để kiểm tra ít nhất 10 chữ N liên tiếp.

[XEM DEMO](http://freetuts.net/editor.html?id=420)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var pattern = /n{10,}/i;  if (pattern.test("10 chu n la nnnnnnnnnn")) {      document.write('Chuỗi có NHIỀU hơn 10 chữ n');  }  else {      document.write('Chuỗi có ÍT hơn 10 chữ n');  } |

**Bài 2**: Kiểm tra chuỗi có phải là '*freetuts.net*' hay không

Để kiểm tra chính xác chuỗi là '*freetuts.net*' thì bắt buộc ta phải thêm ký tự bắt đầu ^ và kết thúc $ để fix chuỗi lại. Và có một lưu ý rằng ký tự . trong chuỗi '*freetuts.net*' là ký tự đặc biệt vì nó có trong danh sách các biểu thức của RegExp nên bắt buộc ta phải thêm dấu \ đằng trước nó.

[XEM DEMO](http://freetuts.net/editor.html?id=421)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var pattern = /^freetuts\.net$/i;  if (pattern.test("freetuts.net")) {      document.write('Chuỗi freetuts.net');  }  else {      document.write('Không phải chuỗi freetuts.net');  } |

**Bài 3**: Kiểm tra chuỗi là các chữ số và dài 8 ký tự

Để kiểm tra các chữ số ta dùng [0-9] và chiều dài 8 ký tự nên ta dùng {8}.

[XEM DEMO](http://freetuts.net/editor.html?id=422)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var pattern = /^[0-9]{8}$/;  if (pattern.test("12345678")) {      document.write('Các số dài 8 ký tự');  }  else {      document.write('Không phải là số hoặc ngắn hơn 8 ký tự');  } |

**Bài 4**: Kiểm tra chuỗi định dạng mã thẻ cào xxxx-xxxx-xxxx-xxxx và x chính là các chữ số.

Ta sử dụng [0-9] để kiểm tra là chữ số. Và các cặp xxxx có chiều dì là 4 nên ta sử dụng {4}.

[XEM DEMO](http://freetuts.net/editor.html?id=423)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var pattern = /^[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4}$/;  if (pattern.test("1234-1232-1321-2312")) {      document.write('Mã thẻ hợp lệ');  }  else {      document.write('Mã thẻ không hợp lệ');  } |

**3. Lời kết**

Với bảng danh sách các biểu thức RegExp này bạn có thể giải mọi bài toán rồi đấy. Tuy nhiên để giải được các bài phức tạp thì đòi hỏi bạn phải thực sự rành rỏi và biết các vận dụng đúng chỗ. Ngoài các biểu thức này ra còn có một số khái niệm như Condition, Lookahead, Lookbehind nhưng mình sẽ không trình bày ở serie này mà nó nằm ở serie [Regular Expression tron PHP](http://freetuts.net/tag/regular-expression).

Bài tiếp theo chúng ta sẽ tìm hiểu một số hàm dùng để xử lý chuỗi RegExp trong Javascript.