


BÀI 6 – ĐỐI TƯỢNG TRONG JAVASCRIPT

6.1. Giới thiệu về đối tượng

Về mặt định nghĩa, một đối tượng (object) là một danh sách các thành phần mà trong đó mỗi thành phần là một cặp tên và giá trị của nó. Trong đó giá trị có thể là các kiểu dữ liệu cơ bản, các hàm hay cũng có thể là một đối tượng khác (được coi là kiểu dữ liệu phức hợp)

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

Như trên ta đã thấy, đối tượng tạo thành bởi thuộc tính và phương thức của nó trong đó thuộc tính để mô tả đối tượng còn phương thức để thể hiện các hành động. Như vậy cách truy xuất vào thuộc tính và phương thức cũng thể hiện khác nhau.

Có 2 cách để truy cập đến thuộc tính đối tượng để lấy giá trị thuộc tính hoặc gán giá trị vào thuộc tính, ví dụ:

objectName.propertyName //Cách 1

objectName['propertyName'] //Cách 2 Cách truy xuất vào thuộc tính

Ví dụ :

```
var person = {  
    name: "John", age: 31,  
    favColor: "green", height: 183  
};  
  
var x = person.age;  
var y = person['age'];
```

Một đối tượng ngoài các thuộc tính ra nó còn chứa hàm gọi là phương thức, ví dụ truy cập một hàm. Cách truy xuất vào phương thức của đối tượng

objectName.methodName()

Ví dụ :

```
name = person.fullName();
```

6.2. Cách tạo đối tượng trong Javascript

Tạo và sử dụng các đối tượng trong JavaScript gần giống với các khái niệm trong lập trình hướng đối tượng, như thuộc tính, phương thức. Các biến trong JavaScript chứa các dữ liệu, các đối tượng cũng tương tự như vậy, nhưng nó chứa được nhiều giá trị.

Cách 1: tạo bằng phương pháp cố định - Tạo đối tượng và khởi tạo luôn các thuộc tính cần có

Ví dụ :

```
var person = {  
    name: "John",  
    age: 42,  
    favColor: "green"  
};
```

Cách 2: Khởi tạo bằng hàm tạo - Khai báo một hàm gọi là hàm tạo rồi tạo ra đối tượng bằng cú pháp new hamtao(). Trong hàm tạo hoặc các hàm thuộc đối tượng, sử dụng từ khóa *this* để tham khảo đến đối tượng, thông qua nó truy cập các thuộc tính (ý nghĩa của this giống this trong Java, Php, C# ...)

Ví dụ :

```
function person(name, age, color) { // Hàm khởi tạo  
    this.name = name; // this tham khảo đến đối tượng cần tạo  
    this.age = age;  
    this.favColor = color;  
}  
  
var p1 = new person("John", 42, "green"); // tạo đối tượng  
var p2 = new person("Amy", 21, "red");    // tạo đối tượng  
document.write(p1.age);                    // Outputs 42  
document.write(p2.name);                  // Outputs "Amy"
```

6.3. Các đối tượng nội tại trong JavaScript

Như đã nói JavaScript là ngôn ngữ lập trình *dựa trên đối tượng*, nhưng không *hướng đối tượng* bởi vì nó không hỗ trợ các lớp cũng như tính thừa kế. Có thể thấy mọi thứ trong javascript đều có thể là đối tượng (trừ kiểu dữ liệu nguyên thủy) như:

- Kiểu boolean có thể là đối tượng nếu được khai báo với từ khóa **new**.
- Kiểu số có thể là đối tượng nếu được khai báo với từ khóa **new**.
- Kiểu xâu ký tự có thể là đối tượng nếu được khai báo với từ khóa **new**.
- Ngày tháng (Dates).
- Toán học (Maths).
- Biểu thức thông thường (Regular expressions).
- Mảng.
- Hàm.

Các đối tượng nội tại trong JavaScript bao gồm đối tượng Math, Date, String.

6.3.1. Đối tượng Math

Đối tượng Math là đối tượng nội tại trong JavaScript. Các thuộc tính của đối tượng này chứa nhiều hằng số toán học, các hàm toán học, lượng giác phổ biến. Đối tượng Math không có chương trình xử lý sự kiện.

Việc tham chiếu tới **number** trong các phương thức có thể là số hay các biểu thức được đánh giá là số hợp lệ.

Các thuộc tính:

- | | |
|-----------|--|
| - E | - Hằng số Euler, khoảng 2,718. |
| - LN2 | - logarit tự nhiên của 2, khoảng 0,693. |
| - LN10 | - logarit tự nhiên của 10, khoảng 2,302. |
| - LOG2E | - logarit cơ số 2 của e, khoảng 1,442. |
| - PI | - Giá trị của π , khoảng 3,14159. |
| - SQRT1_2 | - Căn bậc 2 của 0,5, khoảng 0,707. |
| - SQRT2 | - Căn bậc 2 của 2, khoảng 1,414. |

Các phương thức:

- Math.abs (*number*) - Trả lại giá trị tuyệt đối của *number*.
- Math.acos (*number*) - Trả lại giá trị arc cosine (theo radian) của *number*. Giá trị của *number* phải nằm giữa -1 và 1.
- Math.asin (*number*) - Trả lại giá trị arc sine (theo radian) của *number*. Giá trị của *number* phải nằm giữa -1 và 1.
- Math.atan (*number*) - Trả lại giá trị arc tan (theo radian) của *number*.
- Math.ceil (*number*) - Trả lại số nguyên nhỏ nhất lớn hơn hoặc bằng *number*.
- Math.cos (*number*) - Trả lại giá trị cosine của *number*.
- Math.exp (*number*) - Trả lại giá trị e^{number} , với e là hằng số Euler.

- `Math.floor (number)` - Trả lại số nguyên lớn nhất nhỏ hơn hoặc bằng *number*.
- `Math.log (number)` - Trả lại logarit tự nhiên của *number*.
- `Math.max (num1,num2)` - Trả lại giá trị lớn nhất giữa *num1* và *num2*.
- `Math.min (num1,num2)` - Trả lại giá trị nhỏ nhất giữa *num1* và *num2*.
- `Math.pos (base,exponent)` - Trả lại giá trị base lũy thừa *exponent*.
- `Math.random (r)` - Trả lại một số ngẫu nhiên giữa 0 và 1. Phuong thức này chỉ thực hiện được trên nền tảng UNIX.
- `Math.round (number)` - Trả lại giá trị của *number* làm tròn tới số nguyên gần nhất.
- `Math.sin (number)` - Trả lại sin của *number*.
- `Math.sqrt (number)` - Trả lại căn bậc 2 của *number*.
- `Math.tan (number)` - Trả lại tang của *number*.

6.3.2. Đối tượng Date

Đối tượng Date là đối tượng có sẵn trong JavaScript. Nó cung cấp nhiều phương thức có ích để xử lý về thời gian và ngày tháng. Đối tượng Date không có thuộc tính và chương trình xử lý sự kiện.

Phần lớn các phương thức date đều có một đối tượng Date đi cùng. Các phương thức giới thiệu trong phần này sử dụng đối tượng Date *dateVar*, ví dụ:

```
dateVar = new Date ('August 16, 1996 20:45:04');
```

Các phương thức:

- `dateVar.getDate()` - Trả lại ngày trong tháng (1-31) cho *dateVar*.
- `dateVar.getDay()` - Trả lại ngày trong tuần (0=chủ nhật,...6=thứ bảy) cho *dateVar*.
- `dateVar.getHours()` - Trả lại giờ (0-23) cho *dateVar*.
- `dateVar.getMinutes()` - Trả lại phút (0-59) cho *dateVar*.
- `dateVar.getSeconds()` - Trả lại giây (0-59) cho *dateVar*.
- `dateVar.getTime()` - Trả lại số lượng các mili giây từ 00:00:00 ngày 1/1/1970.
- `dateVar.getTimeZoneOffset()` - Trả lại độ dịch chuyển bằng phút của giờ địa phương hiện tại so với giờ quốc tế GMT.
- `dateVar.getYear()`-Trả lại năm cho *dateVar*.
- `Date.parse (dateStr)` - Phân tích chuỗi *dateStr* và trả lại số lượng các mili giây tính từ 00:00:00 ngày 01/01/1970.
- `dateVar.setDay(day)` - Đặt ngày trong tháng là *day* cho *dateVar*.
- `dateVar.setHours(hours)` - Đặt giờ là *hours* cho *dateVar*.
- `dateVar.setMinutes(minutes)` - Đặt phút là *minutes* cho *dateVar*.
- `dateVar.setMonths(months)` - Đặt tháng là *months* cho *dateVar*.

- *dateVar.setSeconds(seconds)* - Đặt giây là *seconds* cho *dateVar*.
- *dateVar.setTime(value)* - Đặt thời gian là *value*, trong đó *value* biểu diễn số lượng mili giây từ 00:00:00 ngày 01/01/1970.
- *dateVar.setYear(years)* - Đặt năm là *years* cho *dateVar*.
- *dateVar.toGMTString()* - Trả lại chuỗi biểu diễn *dateVar* dưới dạng GMT.
- *dateVar.toLocaleString()* - Trả lại chuỗi biểu diễn *dateVar* theo khu vực thời gian hiện thời.
- *Date.UTC (year, month, day [,hours] [,minutes] [,seconds])* - Trả lại số lượng mili giây từ 00:00:00 01/01/1970 GMT.

6.3.3. Đối tượng String

Đối tượng String là đối tượng được xây dựng nội tại trong JavaScript cung cấp nhiều phương thức thao tác trên chuỗi. Đối tượng này có thuộc tính duy nhất là độ dài (length) và không có chương trình xử lý sự kiện.

Các phương thức:

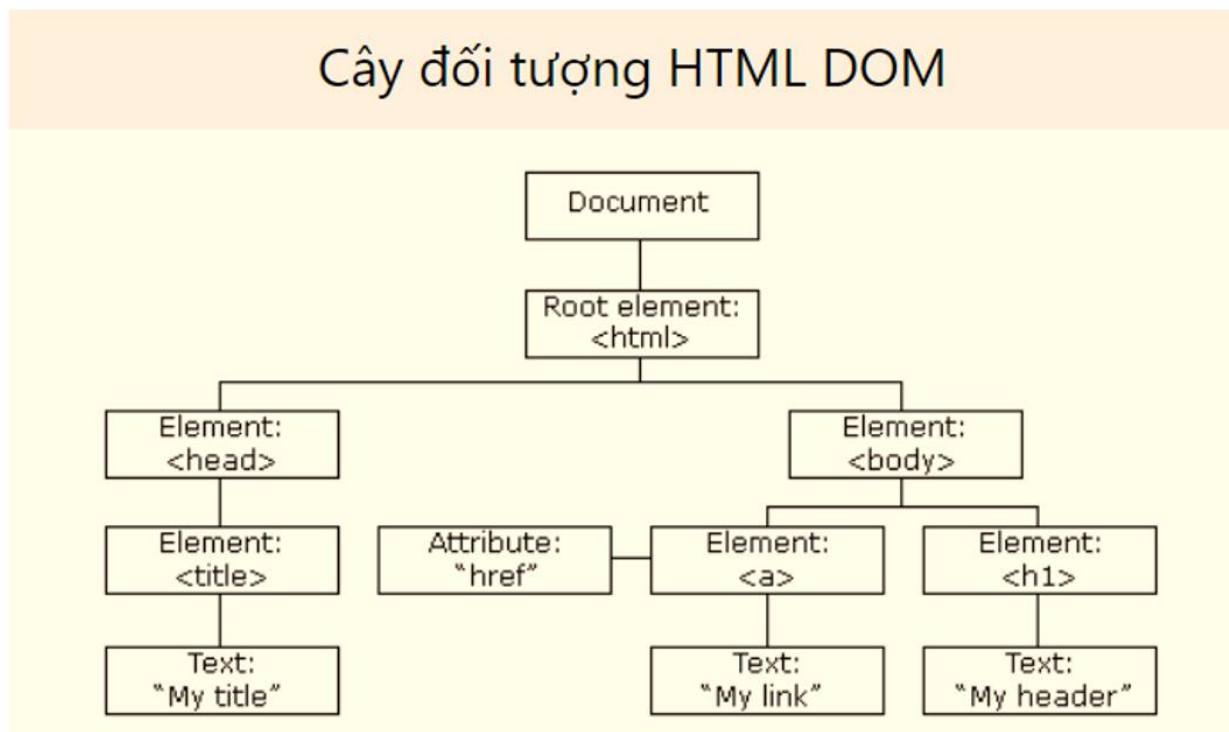
- *str.anchor (name)* - Được sử dụng để tạo ra thẻ <A> (một cách động). Tham số *name* là thuộc tính NAME của thẻ <A>.
- *str.big()* - Kết quả giống như thẻ <BIG> trên chuỗi *str*.
- *str.blink()* - Kết quả giống như thẻ <BLINK> trên chuỗi *str*.
- *str.bold()* - Kết quả giống như thẻ <BOLD> trên chuỗi *str*.
- *str.charAt(a)* - Trả lại ký tự thứ *a* trong chuỗi *str*.
- *str.fixed()* - Kết quả giống như thẻ <TT> trên chuỗi *str*.
- *str.fontcolor()* - Kết quả giống như thẻ <FONTCOLOR = *color*>.
- *str.fontSize(size)* - Kết quả giống như thẻ <FONTSIZE = *size*>.
- *str.indexOf(srchStr [,index])* - Trả lại vị trí trong chuỗi *str* vị trí xuất hiện đầu tiên của chuỗi *srchStr*. Chuỗi *str* được tìm từ trái sang phải. Tham số *index* có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.
- *str italics()* - Kết quả giống như thẻ <I> trên chuỗi *str*.
- *str.lastIndexOf(srchStr [,index])* - Trả lại vị trí trong chuỗi *str* vị trí xuất hiện cuối cùng của chuỗi *srchStr*. Chuỗi *str* được tìm từ phải sang trái. Tham số *index* có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.
- *str.link(href)* - Được sử dụng để tạo ra một kết nối HTML động cho chuỗi *str*. Tham số *href* là URL đích của liên kết.
- *str.small()* - Kết quả giống như thẻ <SMALL> trên chuỗi *str*.
- *str.strike()* - Kết quả giống như thẻ <STRIKE> trên chuỗi *str*.
- *str.sub()* - Tạo ra một subscript cho chuỗi *str*, giống thẻ <SUB>.
- *str.substring(a,b)* - Trả lại chuỗi con của *str* là các ký tự từ vị trí thứ *a* tới vị trí thứ *b*. Các ký tự được đếm từ trái sang phải bắt đầu từ 0.

- `str.sup()` - Tạo ra superscript cho chuỗi *str*, giống thẻ `<SUP>`.
- `str.toLowerCase()` - Đổi chuỗi *str* thành chữ thường.
- `str.toUpperCase()` - Đổi chuỗi *str* thành chữ hoa.

6.4. BOM và DOM trong Javascript

Các đối tượng liên quan đến trình duyệt (BOM - Browser Object Model). Mỗi browser sẽ có những đối tượng khác nhau nên 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, screen, location, history, navigator, popup, timing, cookies.

Bên cạnh đó DOM (Document Object Model) là các thành phần được tạo ra từ trang HTML do người lập trình xây dựng. Quá trình hình thành DOM được thể hiện theo cấu trúc hình cây, thông qua đó có thể thay đổi, thêm hoặc xóa các phần tử HTML



6.4.1. Đối tượng window

Đối tượng window là đối tượng được hỗ trợ trên tất cả các trình duyệt, có thể hiểu nó đại diện cho cửa sổ trình duyệt. Tất cả các đối tượng có chứa trong javascript, các hàm, các biến, đều tự động trở thành thuộc tính (thành viên) của đối tượng window.

Biến là thuộc tính, hàm là phương thức của đối tượng window, ngay cả đối tượng tài liệu (DOM HTML) là một thuộc tính của đối tượng window.

Ví dụ:

```
window.document.getElementById("header");
```

được giống như:

```
document.getElementById("header");
```

Các thuộc tính: có 2 thuộc tính để xác định kích thước của cửa sổ và trả về giá trị pixel đó là *window.innerHeight* - chiều cao bên trong cửa sổ và *window.innerWidth* - chiều rộng bên trong cửa sổ. Kích thước này không tính thanh công cụ (toolbars) và thanh cuộn (scrollbars).

Các phương thức:

- `window.alert("message")` - Hiện thị hộp hội thoại với chuỗi "message" và nút OK.
- `window.close()` - Đóng cửa sổ `windowReference`.
- `window.confirm("message")` - Hiện thị hộp hội thoại với chuỗi "message", nút OK và nút Cancel. Trả lại giá trị True cho OK và False cho Cancel.
- `window.open()` - Mở cửa sổ mới.
- `window.prompt("message" [, "defaultInput"])` - Mở một hộp hội thoại để nhận dữ liệu vào trường text.
- `window.moveTo()` - di chuyển cửa sổ hiện tại.
- `window.resizeTo()` - thay đổi kích thước cửa sổ hiện tại.

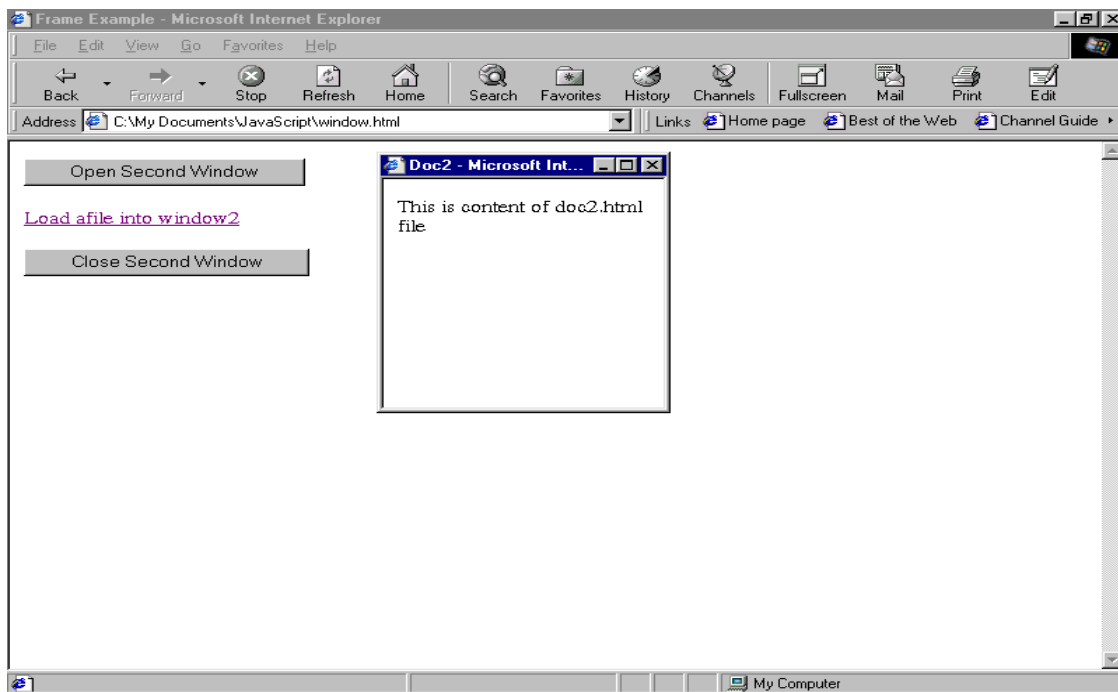
Ví dụ: Thể hiện một biểu mẫu gồm 3 thành phần 2 nút ấn và liên kết, nút thứ nhất để mở một cửa sổ rộng, liên kết để tải file `doc2.html` xuống cửa sổ mới vừa mở ra, nút thứ 2 dùng để đóng cửa sổ thứ hai vừa mở ra. Ví dụ này lưu vào file *window.html*:

```
<HTML>
<HEAD>
<TITLE>Frame Example </TITLE>
</HEAD>
<BODY>
<FORM>
<INPUT      TYPE="button"      VALUE="Open      Second      Window"
onClick="msgWindow=window.open('', 'window2', 'resizable=no, width=200, height=200') ">
<P>
<A HREF="doc2.html" TARGET="window2">
```

```

Load a file into window2 </A>
</P>
<INPUT TYPE="button" VALUE="Close Second Window"
        onClick="msgWindow.close()">
</FORM>
</BODY>
</HTML>

```



Hình 40: Minh họa cho đối tượng cửa sổ

Các chương trình xử lý sự kiện

- onLoad - Xuất hiện khi cửa sổ kết thúc việc tải.
- onUnload - Xuất hiện khi cửa sổ được loại bỏ.

6.4.2. Đối tượng navigator

Đối tượng window.navigator chứa thông tin về trình duyệt của người truy cập, khi viết ta có thể bỏ qua tiền tố window. Ví dụ như navigator.platform

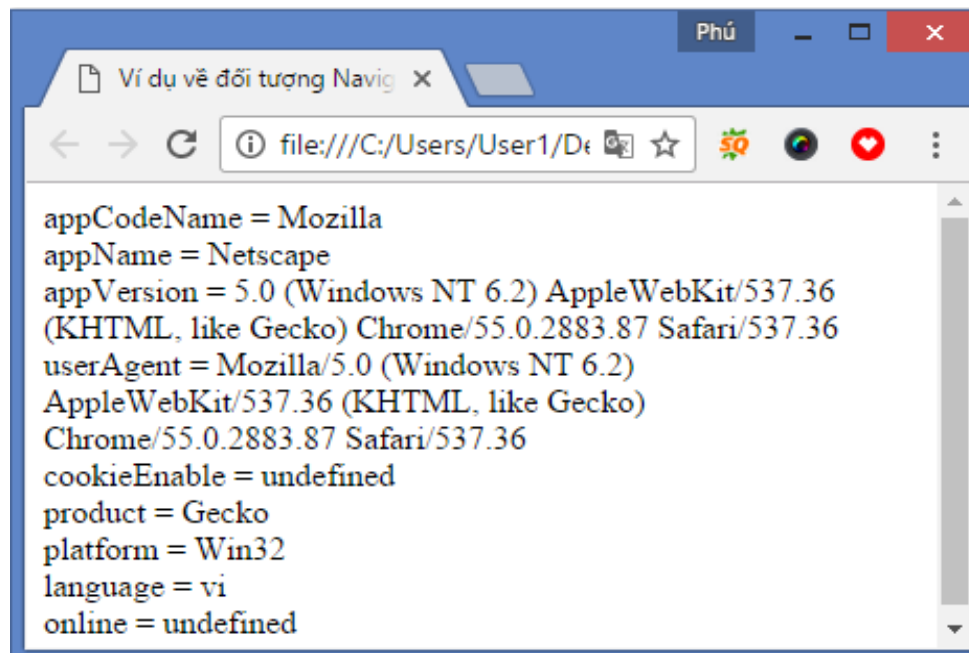
Các thuộc tính	Mô tả
<i>appCodeName</i>	Xác định tên mã nội tại của trình duyệt (Atlas).
<i>AppName</i>	Xác định tên trình duyệt.

<i>AppVersion</i>	Xác định thông tin về phiên bản của đối tượng navigator.
<i>userAgent</i>	Xác định header của user – agent được gửi tới máy chủ
<i>cookieEnabled</i>	Trả về giá trị đúng nếu tệp tinh cookie được kích hoạt
<i>product</i>	Trả về tên các công cụ của trình duyệt
<i>platform</i>	Trả về nền tảng của trình duyệt (hệ điều hành)
<i>language</i>	Trả về ngôn ngữ của trình duyệt
<i>online</i>	Trả về giá trị đúng nếu trình duyệt đang sử dụng

Phương thức `javaEnable()` trả về giá trị đúng nếu java đang được bật
Ví dụ sau sẽ hiển thị các thuộc tính của đối tượng navigator

```
<HTML>
<HEAD>
<TITLE> Ví dụ về đối tượng Navigator </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
document.write("appCodeName   =   "+navigator.appCodeName   +
"<BR>");
document.write("appName = "+navigator.appName + "<BR>");
document.write("appVersion   =   "+navigator.appVersion   +
"<BR>");
document.write("userAgent     =     "+navigator.userAgent     +
"<BR>");
document.write("cookieEnable = "+navigator.cookieEnable +
"<BR>");
document.write("product = "+navigator.product + "<BR>");
document.write("platform = "+navigator.platform + "<BR>");
document.write("language = "+navigator.language + "<BR>");
document.write("online = "+navigator.online + "<BR>");
```

```
document.write("javaEnable = "+navigator.javaEnable());
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```



Hình 41: Ví dụ đối tượng Navigator

6.4.3. Đối tượng location

Các thuộc tính của đối tượng location duy trì các thông tin về URL của document hiện thời. Ví dụ: [http:// www.abc.com/ chap1/page2.html#topic3](http://www.abc.com/chap1/page2.html#topic3).

Các thuộc tính:

- hostname - Tên của host và domain (ví dụ www.abc.com).
- href - Toàn bộ URL cho document hiện tại.
- pathname - Phần đường dẫn của URL (ví dụ /chap1/page2.html).
- protocol - Giao thức được sử dụng (cùng với dấu hai chấm) (ví dụ http:).

Phương thức: location.assign() được sử dụng để tải lại một địa chỉ URL được xác định trong assign. Ví dụ: location.assign("http://www.fit-hau.edu.vn").

6.4.4. Đối tượng history

Đối tượng history chứa đựng lịch sử của trình duyệt, để đảm bảo tính riêng tư của người dùng javascript cũng hạn chế việc truy cập vào đối tượng này. Có hai phương thức được hỗ trợ chính bao gồm:

- `history.back()`: Được thực hiện tương tự như nút quay lại (back) trên trình duyệt, khi gọi phương thức này nó sẽ tải lại địa chỉ URL trước đó trong danh sách lịch sử.
- `history.forward()`: Được thực hiện tương tự như nút chuyển tiếp (forward) trên trình duyệt, khi gọi phương thức này nó sẽ tải lại địa chỉ URL kết tiếp trong danh sách lịch sử.

Ví dụ:

```
<html><head>

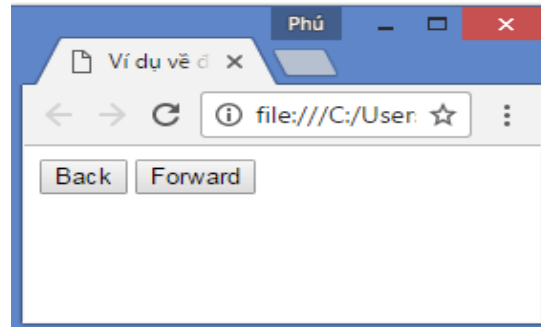
<script>

function goForward() {
window.history.forward()
}

function goBack() {
window.history.back()
}

</script></head>

<body>
<input type="button" value="Back" onclick="goBack()" >
<input type="button" value="Forward" onclick="goForward()" >
</body>
</html>
```



Hình 42. Ví dụ về đối tượng history

6.4.5. Đối tượng document

Đối tượng document đại diện cho trang tài liệu HTML, nếu muốn truy cập vào bất kỳ một thành phần nào của HTML phải luôn thực hiện bắt đầu với đối tượng này.

Các đối tượng anchor, forms, history, links là thuộc tính của đối tượng document. Không có các chương trình xử lý sự kiện cho các frame. Sự kiện onLoad và onUnload là cho đối tượng window.

Các thuộc tính:

- `alinkColor` - Giống như thuộc tính `ALINK`.
- `anchor` - Mảng tất cả các `anchor` trong document.
- `bgColor` - Giống thuộc tính `BGCOLOR`.
- `cookie` - Sử dụng để xác định cookie.
- `fgColor` - Giống thuộc tính `TEXT`.
- `forms` - Mảng tất cả các form trong document.
- `lastModified` - Ngày cuối cùng văn bản được sửa.
- `linkColor` - Giống thuộc tính `LINK`.
- `links` - Mảng tất cả các link trong document.
- `location` - URL đầy đủ của văn bản.
- `referrer` - URL của văn bản gọi nó.
- `title` - Nội dung của thẻ `<TITLE>`.
- `vlinkColor` - Giống thuộc tính `VLINK`.

Các phương thức:

- Tìm kiếm các thành phần HTML:
 - o `document.getElementById(id)`: Thực hiện tìm kiếm các thành phần HTML theo id được khai báo trong thẻ HTML.
 - o `document.getElementsByTagName(name)`: Tìm các thành phần bởi tên thẻ HTML.
 - o `document.getElementsByClassName(name)`: Tìm các thành phần HTML bởi tên lớp (class name) được khai báo trong HTML.
- Thay đổi các thành phần HTML:
 - o `element.innerHTML = new html content`: thay đổi nội dung bên trong một thành phần HTML.
 - o `element.attribute = new value`: thay đổi giá trị thuộc tính của thẻ HTML.
 - o `element.setAttribute(attribute, value)`: thay đổi giá trị thuộc tính của thẻ HTML.
 - o `element.style.property = new style`: thay đổi thuộc tính style của một thẻ HTML.
- Thêm mới hoặc xóa bỏ một thành phần HTML
 - o `document.createElement(element)`: tạo một thẻ HTML.
 - o `document.removeChild(element)`: bỏ một thẻ HTML.
 - o `document.appendChild(element)`: thêm một thẻ HTML.
 - o `document.replaceChild(element)`: thay thế một thẻ HTML.
 - o `document.write(text)`: viết các thẻ HTML theo dạng chuỗi.
- Thêm trình xử lý sự kiện (adding event handlers):

- `document.getElementById(id).onclick = function(){code}`: cho phép thêm một đoạn mã lệnh xử lý sự kiện cho một thành phần HTML thông qua sự kiện nhấp chuột (click event).

6.4.6. *Đối tượng forms*

Đối tượng form được tạo ra bởi thẻ `<form>` trong HTML và để truy xuất vào các thành phần của một form có thể thực hiện phương thức **getElementById**. Để khởi tạo đối tượng form bằng javascript có thể sử dụng phương thức **createElement** của đối tượng `document`. Ví dụ: `document.createElement("FORM")`.

Tập hợp các đối tượng của form (form object collections): Đó là các phần tử (elements) tham gia vào form, chúng xuất hiện theo thứ tự trình bày trong trang HTML. Để truy xuất vào từng thành phần sẽ phải thực hiện thông qua chỉ số (index), việc truy xuất được thực hiện theo cú pháp: **formObject.elements**. Để lấy số lượng thành phần thuộc form có thể sử dụng thuộc tính *length*.

Các phương thức: Gồm 2 phương thức chính là reset và submit, các phương thức này thực hiện tương tự như nút ấn reset và submit của thẻ `<form>`.

Các thuộc tính: Ngoài các phương thức kể trên đối tượng form còn các thuộc tính là thuộc tính của thẻ `<form>` bao gồm:

- `acceptCharset`: Thiết lập hoặc trả về giá trị của thuộc tính `accept - charset` của thẻ `<form>`.
- `action`: Thiết lập hoặc trả về giá trị `action` của thẻ `<form>`.
- `autocomplete`: Thiết lập hoặc trả về giá trị `autocomplete` của thẻ `<form>`.
- `encoding`: thiết lập hoặc trả về giá trị `encoding` của thẻ `<form>`.
- `enctype`: thiết lập hoặc trả về giá trị `enctype` của thẻ `<form>`.
- `method`: thiết lập hoặc trả về giá trị `method` của thẻ `<form>`.
- `name`: thiết lập hoặc trả về giá trị `name` của thẻ `<form>`.
- `noValidate`: Thiết lập hoặc trả về dữ liệu mà không cần kiểm soát trong quá trình gửi đi.
- `target`: Thiết lập hoặc trả về giá trị `target` của thẻ `<form>`.

Ngoài các thuộc tính trên thì đối tượng form còn rất nhiều thuộc tính khác được sử dụng từ các thuộc tính của thẻ `<form>` và các thành phần trên form. Các thành phần này bao gồm thẻ `<input>`, `<select>` và `<textarea>`.

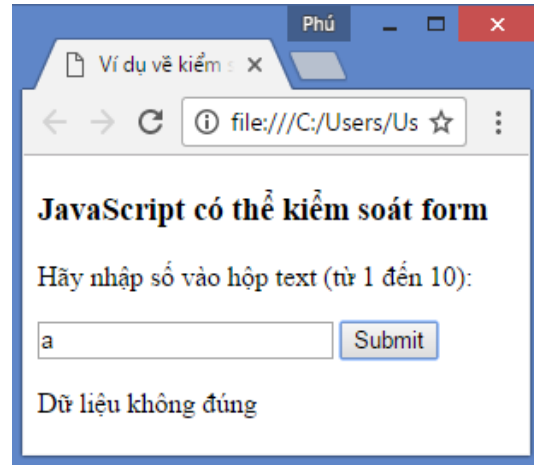
Trong javascript, đối tượng form khi được sử dụng thường để kiểm soát dữ liệu trước khi gửi đi. Nếu một trường trên form bị trống, hàm kiểm soát sẽ thông báo và trả về kết quả sai nhằm ngăn chặn việc gửi các dữ liệu sai đi. Javascript có thể kiểm soát việc nhập số vào hộp text.

Ví dụ:

```

<HTML><Head>
<TITLE> Ví dụ về kiểm soát dữ liệu trên form </TITLE>
</head>
<body>
<h3>JavaScript có thể kiểm soát form</h3>
<p>Hãy nhập số vào hộp text (từ 1 đến 10):</p>
<input id="numb">
<button type="button" onclick="myFunction()"> Submit
</button>
<p id="demo"></p>
<script>
function myFunction() {
    var x, text;
    // Nhận giá trị từ hộp text với id="numb"
    x = document.getElementById("numb").value;
    // Nếu x không phải là số và không trong khoảng từ 1 đến 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Dữ liệu không đúng";
    } else {
        text = "Dữ liệu đúng";
    }
    document.getElementById("demo").innerHTML = text;
}
</script>
</body>

```



Hình 43. Kiểm soát dữ liệu số

</HTML>

Để kiểm soát quá trình nhập liệu trên form và đặc biệt qua hộp text, HTML có thể tự kiểm soát việc để trống thông qua thuộc tính **required** của thẻ <INPUT>. Ví dụ:

```
<form action="demo_form.php" method="post">
  <input type="text" name="fname" required><br>

  <input type="password" name="pwd" required><br>
  <input type="submit" value="Submit">
</form>
```

Kiểm soát dữ liệu là quá trình đảm bảo rằng dữ liệu do người dùng nhập vào chính xác, hợp lý và hữu ích. Nhiệm vụ kiểm soát dữ liệu chính bao gồm điền đầy đủ thông tin vào tất cả các trường, điền đúng định dạng ngày tháng, điền đúng kiểu dữ liệu số. Kiểm soát việc nhập liệu có thể được thực hiện bằng nhiều cách khác nhau. Nếu kiểm soát dữ liệu được thực hiện phía máy chủ web thì dữ liệu sẽ được kiểm soát sau khi dữ liệu đã gửi tới máy chủ. Ngược lại, nếu việc kiểm soát được thực hiện phía máy trạm dữ liệu sẽ được kiểm soát trước khi dữ liệu được gửi tới máy chủ.