1. **TỔNG QUAN VỀ JAVASCRIPT**
   1. **Giới thiệu**

JavaScript là một ngôn ngữ kịch bản dựa trên đối tượng nhằm phát triển các ứng dụng Internet chạy trên phía client và phía server.

Mục tiêu của JavaScript là nhằm cung cấp cho các nhà phát triển Web một số khả năng và quyền điều khiển chức năng cho trang Web. Mã Javascript có khả năng nhúng trong tài liệu HTML để điều khiển nội dung của trang Web và kiểm tra sự hợp lệ của dữ liệu mà người dùng nhập vào. Khi một trang hiển thị trong trình duyệt, các câu lệnh được trình duyệt thông dịch và thực thi.

* 1. **Hiệu ứng và quy tắc Javascript**

Chúng ta có thể sử dụng JavaScript để:

* Tương tác với người dùng. Chúng ta có thể viết mã để đáp lại các sự kiện. Các sự này sẽ có thể phát sinh bởi người dùng, nhấp chuột hay được phát sinh từ hệ thống, định lại kích thước của trang và v.v.
* Thay đổi nội dung động. Mã JavaScript có thể dùng để thay đổi nội dung và vị trí các phần tử một cách động trên một trang nhằm đáp lại sự tương tác với người dùng.
* Kiểm tra tính hợp lệ dữ liệu. Chúng ta có thể viết mã nhằm kiểm tra tính hợp lệ của dữ liệu do người dùng nhập vào trước khi nó được gửi lên Web server để xử lý.
  1. **Nhúng JavaScript vào trong trang Web**

Chúng ta có thẻ chèn các lệnh JavaScritp vào trong một tài liệu HTML theo những cách sau đây:

* Nhúng các câu lệnh trực tiếp vào trong tài liệu bằng cách sử dụng thẻ <SCRIPT>

<script

<!-- JavaScript statements; //-->

</script>

* Liên kết file nguồn JavaScript với tài liệu HTML

<script language="JavaScript" src="myScript.js"> x

</script>

* 1. **Biến**

Khái báo 1 biến:

var tên\_biến = giá\_trị ;

hoặc **:**

tên\_biến = giá\_trị

* 1. **Các kiểu dữ liệu**

|  |  |
| --- | --- |
| **Kiểu dữ liệu** | **Mô tả** |
| Numbers | Các số nguyên hoặc số thực. Ví dụ: 487 hoặc 25.95 |
| Logical hoặc Boolean | true hoặc false |
| Strings | “Hello World” |
| Null | Keyword đặc biệt chỉ giá trị null. |

Khi khai báo một biến thì không cần phải chỉ ra kiểu dữ liệu của nó. Các kiểu dữ liệu sẽ tự động thay đổi nếu cần thiết trong quá trình thực thi script.

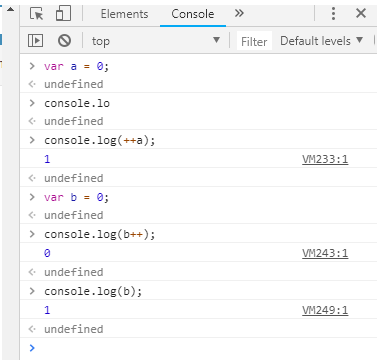
Hàm chuyển đổi kiều dữ liệu: *parseInt()* và *parseFloat()* có chức năng chuyển đổi các chuỗi thành các số nguyên hoặc các số thực

* 1. **Các toán tử**

1. *Toán tử số học*

|  |  |
| --- | --- |
| **Toán tử** | **Mô tả** |
| + | Phép cộng |
| - | Phép trừ |
| / | Phép chia |
| % | Phép chia lấy số dư |
| ++ | ++x sẽ trả về giá trị của x sau khi tăng.  x++ sẽ trả về giá trị của x trước khi tăng. |
| -- | --x sẽ trả về giá trị của x sau khi giảm.  x-- sẽ trả về giá trị của x trước khi giảm. |

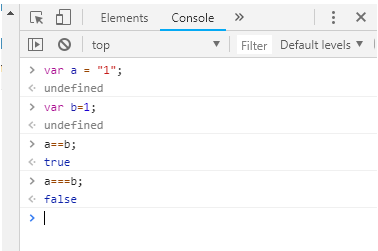
VD phép ++



1. *Toán tử so sánh*

|  |  |
| --- | --- |
| **Toán tử** | **Mô tả** |
| == | Bằng. Trả về giá trị true nếu các toán hạng bằng nhau. |
| != | Không bằng. Trả về giá trị true nếu các toán hạng không bằng nhau. |
| > | Lớn hơn. Trả về giá trị true nếu toán hạng trái lớn hơn toán hạng phải |
| >= | Lớn hơn hoặc bằng. Trả về giá trị true nếu toán hạng trái lớn hơn hoặc bằng toán hạng phải. |
| < | Nhở hơn. Trả về giá trị true nếu toán hạng trái nhỏ hơn toán hoạng phải |
| <= | Nhỏ hơn hoặc bằng. Trả vè giá trị true nếu toán hạng trái nhỏ hơn hoặc bằng toán hạng phải. |

VD: sự khác nhau giữa == và ===



1. *Toán tử logic*

|  |  |  |
| --- | --- | --- |
|  |  |  |
| And ( &&) | expr1 && expr2 | Trả về giá trị của expr1 nếu nó là false. Nếu không thì nó trả về giá trị của expr2. |
| Or ( ||) | expr1 || expr2 | Trả về giá trị của expr1 nếu nó là true. Nếu không thì nó sẽ trả về giá trị của expr2 |
| Not (!) | !expr | Trả về giá trị false nếu biểu thức đúng và trả về giá trị true nếu biểu thức sai. |

1. *Toán tử gán*

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Ví dụ** | **Tương tự** |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x – y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |
| <<= | x <<= y | x = x << y |
| >>= | x >>= y | x = x >> y |
| >>>= | x >>>= y | x = x >>> y |
| &= | x &= y | x = x & y |
| ^= | x ^= y | x = x ^ y |
| |= | x |= y | x = x | y |
| \*\*= | x \*\*= y | x = x \*\* y |

1. *Toán tử chuỗi*

Toán tử chuỗi nhận hai toán hạng và tạo ra một chuỗi mới bằng cách ghép hai chuỗi ban đầu lại với nhau.

* 1. **Các lệnh cơ bản trong Js**

1. *Lệnh if…else*

Lệnh if: là một lệnh điều khiển cơ bản mà cho phép JavaScript điều khiển luồng và thực thi các lệnh theo điều kiện. một biểu thức JavaScript được ước lượng. Nếu giá trị kết quả là true, lệnh đã cung cấp được thực thi. Nếu biểu thức là false, thì không có lệnh nào sẽ được thực thi. Hầu hết mọi lần, sẽ sử dụng các toán tử so sánh trong khi điều khiển luồng.

Lệnh if...else: là một mẫu lệnh điều khiển khác cho phép JavaScript thực thi các lệnh trong một cách được điều khiển hơn. Ở đây, biểu thức JavaScript được ước lượng. Nếu giá trị kết quả là true, lệnh đã cung cấp trong khối if được thực thi. Nếu biểu thức là false, thì khi đó lệnh đã cho trong khối else được thực thi.

Lệnh if...else if...: là một mẫu lệnh điều khiển **if…else** nâng cao mà cho phép JavaScript tạo một quyết định chính xác trong một số điều kiện. Nó là một loạt các lệnh **if**, mỗi **if** là một phần của mệnh đề **else**của lệnh trước. Các lệnh được thực thi trên cơ sở điều kiện true, nếu không có điều kiện nào true, thì khi đó khối **else** được thực thi.

1. *Lệnh Switch Case*

switch (expression)

{

case condition 1: statement(s)

break;

case condition 2: statement(s)

break;

...

case condition n: statement(s)

break;

default: statement(s)

}

1. *Vòng lặp while*

Vòng lặp while là để thực thi một lệnh hoặc khối code lặp đi lặp lại miễn là expression – biểu thức là true. Một khi biểu thức trở thành false, vòng lặp kết thúc.

while (expression){

Statement(s) to be executed if expression is true

}

1. *Vòng lặp for*

for (initialization; test condition; iteration statement){

Statement(s) to be executed if test condition is true

}

* 1. **Js coding convention**
* Quy tắc cú pháp trong JS:
* Dùng Caps. JavaScript phân biệt chữ hoa chữ thường.
* Dùng Pairs. Trong JavaScript, luôn luôn có cặp ký hiệu mở và đóng. Lỗi sẽ xuất hiện khi bỏ sót hoặc đặt sai một trong hai ký hiệu này.
* Dùng Spaces (các ký tự trắng). Như HTML, JavaScript thường bỏ qua ký tự trắng. Trong JavaScript, ta có thể thêm vào các ký tự trắng hoặc các tab giúp cho ta dễ dàng đọc hay sửa các file script.
* Dùng Chú thích (Comments). Các chú thích giúp ta ghi chú về chức năng của đoạn script, thời gian và người tạo ra đoạn script.
* quy tắc đặt tên biến:
* Tên biến phải bắt đầu bằng bằng chữ hoặc kí tự gạch dưới \_, $.
* Tên biến không được bắt đầu bằng số và không chứa các kí tự đặc biệt như &, \*, (, ) …
* Tên biến không được trùng với từ khóa như var, for, if...
* Tên biến phải mang ý nghĩa rõ ràng
* Một số phong cách đặt tên biến và hàm:

**Underscore**

Mỗi từ cách nhau bởi mỗi dấu gạch dưới.

Ví dụ: name\_of\_cat, title\_website, windows\_height...

**Camel Case**

Những chữ cái đầu của mỗi từ đều viết hoa, những chữ cái còn lại đều viết thường.

Ví dụ: NameOfCat, TitleWebsite, WindowsHeight..

Một kiểu style được biến tấu từ Camel Case là những chữ cái đầu tiên viết thường và những chữ sau đó tuân theo Camel Case

Ví dụ: nameOfCat, titileWebsite, windowsHeight.

* Khoảng cách giữa các toán tử:

Có khoảng cách giữa các toán tử (+ – \* / =), và sau dấu phẩy

Vd: var x = y + z;

var values = ["Volvo", "Saab", "Fiat"];

* Thụt đầu dòng:

Nên thụt đầu dòng bằng 4 khoảng cách (4 lần phím space)

Không sử dụng tab để thụt đầu dòng. Mỗi trình duyệt sẽ định nghĩa khoảng cách mỗi tab khác nhau.

* Quy ước về biểu thức

Quy ước chung là kết thúc biểu thức bằng dấu chấm phẩy:

Vd: var values = ["Volvo", "Saab", "Fiat"];

var person = {

    firstName: "John",

    lastName: "Doe",

    age: 50,

    eyeColor: "blue"

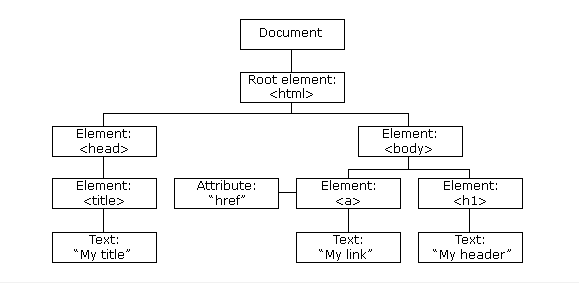
};

Một vài quy ước nữa được đề cập như sau:

* Đặt dấu ngoặc nhọn mở tại cuối dòng.
* Đặt 1 khoảng cách trước khi mở dấu ngoặc nhọn.
* Đặt dấu ngoặc nhọn đóng ở tại dòng mới, và không có khoảng cách.
* Các biểu thức phức tạp (như hàm, vòng lặp hay điều kiện) không kết thúc bằng dấu chấm phẩy.
* Quy ước cho đối tượng:
* Các quy ước chung về việc định nghĩa đối tượng như sau:
* Đặt dấu ngoặc nhọn mở trên cùng 1 dòng với tên đối tượng
* Dùng dấu hai chấm và 1 khoảng cách giữa tên thuộc tính và giá trị của thuộc tính đó
* Sử dụng dấu nháy bao quanh giá trị chuỗi, không dùng dấu nháy cho số
* Không thêm dấu phẩy vào cặp thuộc tính/giá trị cuối cùng của đối tượng đó
* Đặt dấu ngoặc nhọn đóng tại dòng mới, không có khoảng cách
* Luôn luôn kết thúc định nghĩa đối tượng bằng dấu chấm phẩy
* Độ dài của dòng:
* Để dễ đọc code, nên tránh để độ dài đoạn code trên 1 dòng lớn hơn 80 ký tự.
* Có thể ngắt dòng nếu code quá dài, và ngắt sau 1 toán tử hoặc dấu phẩy

1. **DOM**
   1. **HTML DOM trong JavaScript**

Khi trang web được tải, trình duyệt sẽ tạo ra một mô hình đối tượng tài liệu của trang. Mô hình DOM HTML được xây dựng như một cây của đối tượng:



*Hình: Cấu trúc DOM*

Với mô hình đối tượng, JavaScript có được tất cả yếu tố cần thiết để tạo ra HTML động:

* JavaScript có thể thay đổi tất cả các phần tử HTML trong trang
* JavaScript có thể thay đổi tất cả thuộc tính HTML trong trang
* JavaScript có thể thay đổi tất cả các kiểu CSS trong trang
* JavaScript có thể xóa các phần tử và thuộc tính HTML hiện có
* JavaScript có thể thêm các phần tử và thuộc tính HTML mới
* JavaScript có thể phản ứng lại tất cả các sự kiện HTML hiện có trong trang
* JavaScript có thể tạo các sự kiện HTML mới trong trang

HTML DOM là một mô hình đối tượng tiêu chuẩn và giao diện lập trình cho HTML:

Các phần tử HTML như các đối tượng

Các thuộc tính của tất cả các phần tử HTML

Các phương thức để truy cập vào tất cả các phần tử HTML

Các sự kiện cho tất cả các yếu tố HTML

Nói cách khác: HTML DOM là một tiêu chuẩn để làm thế nào để có được, thay đổi, thêm, hoặc xóa các phần tử HTML

* 1. **Đối tượng tài liệu HTML DOM**

### *Tìm các phần tử HTML*

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| document.getElementById(id) | Tìm một phần tử theo id |
| document.getElementsByTagName(name) | Tìm các phần tử theo tên thẻ |
| document.getElementsByClassName(name) | Tìm các phần tử theo tên lớp |

### *Thay đổi phần tử HTML*

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| element.innerHTML =  new html content | Thay đổi HTML bên trong của một phần tử |
| element.attribute = new value | Thay đổi giá trị thuộc tính của một phần tử HTML |
| element.setAttribute(attribute, value) | Thay đổi giá trị thuộc tính của một phần tử HTML |
| element.style.property = new style | Thay đổi định dạng của một phần tử HTML |

### *Thêm và xóa phần tử HTML*

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| document.createElement(element) | Tạo một phần tử HTML |
| document.removeChild(element) | Bỏ một phần tử HTML |
| document.appendChild(element) | Thêm một phần tử HTML |
| document.replaceChild(element) | Thay thế một phần tử HTML |
| document.write(text) | Viết vào luồng đầu ra HTML |

### *Thêm bộ quản lý sự kiện*

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| document.getElementById(id).onclick=function(){code} | Thêm mã trình xử lý sự kiện vào sự kiện onclick |

* 1. **Thay đổi HTML**

Để sửa đổi nội dung của một phần tử HTML là sử dụng thuộc tính innerHTML.

Cú pháp:

document.getElementById(id).innerHTML = new HTML

* 1. **Thay đổi CSS**

Để thay đổi kiểu của một phần tử HTML, sử dụng cú pháp:

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

* 1. **DOM Animations**

Để tạo ra các hình ảnh động HTML với JavaScript thực hiện bằng cách lập trình thay đổi dần theo định dạng của một phần tử. Các thay đổi được gọi bởi một bộ đếm thời gian. Khi khoảng thời gian bộ đếm nhỏ, hoạt ảnh sẽ liên tục.

var id = setInterval(frame, 5);   
function frame() {  
    if (/\* test for finished \*/) {  
        clearInterval(id);  
    } else {  
        /\* code to change the element style \*/    
    }  
}

* 1. **Sự kiện**
* Onfocus, onblur: thường dùng cho <input>, <select> và <a>.

VD:

<script type="text/javascript">

function changeColor(choose) {

var txt = document.getElementById("test");

if(choose==1) { //on focus

txt.style.border="4px solid red";

}

else { //on blur

txt.style.border="4px solid blue";

}

}

</script>

<input type="text" id="test" onFocus="changeColor(1)" onBlur="changeColor(3)">

* onmouseover, onmouseout , onmousedown, onmouseup

VD:

<script type="text/javascript">

function changeColorButton (thamSo){

var nut=document.getElementById("button");

if (thamSo==1) { //hover onmouseover

nut.style.backgroundColor="red";

}

else if (thamSo==2) { // onmouseout

nut.style.backgroundColor="white";

}

else if (thamSo==3) { //down

nut.style.backgroundColor="green";

}

else { //up

nut.style.backgroundColor="yellow";

}

}

</script>

<input type="button" id="button" value="tác dụng vào đây"

onmouseover="changeColorButton(1)"

onmouseout="changeColorButton(2)"

onmousedown="changeColorButton(3)"

onmouseup="changeColorButton(5)">

1. **JSON**
   1. **Giới thiệu**

JSON (JavaScript Object Notation) là một cú pháp để lưu trữ và trao đổi dữ liệu. JSON là văn bản, được viết bằng ký hiệu đối tượng JavaScript.

JSON là văn bản và có thể chuyển đổi bất kỳ đối tượng JavaScript nào thành JSON và gửi JSON đến máy chủ. Ngược lại, cũng có thể chuyển đổi bất kỳ JSON nào nhận được từ máy chủ sang các đối tượng JavaScript.

VD: chuyển dữ liệu:

var myObj = {

name: "John",

age: 31,

city: "New York"

};  
var myJSON = JSON.stringify(myObj);  
window.location = myJSON;

VD: nhận dữ liệu:

var myJSON = '{"

name":"John",

"age":31,

"city":"New York"

}';  
var myObj = JSON.parse(myJSON);  
document.getElementById("demo").innerHTML = myObj.name;

Khi lưu trữ dữ liệu, dữ liệu phải là định dạng nhất định. JSON có thể lưu trữ các đối tượng JavaScript dưới dạng văn bản.

VD:

//Lưu trữ dữ liệu  
myObj = {

name: "John",

age: 31,

city: "New York"

};  
myJSON = JSON.stringify(myObj);  
localStorage.setItem("testJSON", myJSON);  
  
//truy xuất dữ liệu  
text = localStorage.getItem("testJSON");  
obj = JSON.parse(text);  
document.getElementById("demo").innerHTML = obj.name;

* 1. **Cú pháp**

Cú pháp JSON là một tập con của cú pháp JavaScript.

Các quy tắc cú pháp JSON

* Dữ liệu nằm trong cặp key/value

VD: “name” : ”linh”

* Dữ liệu được phân cách bằng dấu phẩy

VD: “name” : ”linh”,

“age” : “22”,

* Đối tượng nằm trong dấu ngoặc nhọn

VD: {

“name” : ”linh”,

“age” : “22”

}

* Mảng nằm trong dấu ngoặc vuông

VD: [

     {

         "name" : "Nguyễn Văn A",

         "age" : "21 tuổi"

     },

     {

         "name" : "Nguyễn Văn B",

         "age" : "22 tuổi"

     },

     {

         "name" : "Nguyễn Văn C",

         "age" : "23 tuổi"

   }

]

Tạo và truy cập một đối tượng JavaScript:

var person = {

"name" : "John",

"age" : 31,

"city" : "New York"

};

document.getElementById("demo").innerHTML = person.name;

* 1. **Các kiểu dữ liệu trong JSON**

Trong JSON, giá trị phải là một trong các loại dữ liệu sau:

* một chuỗi

{ "name" : "John" }

* một số

{ "age" : 30 }

* một đối tượng (đối tượng JSON)

{

"employee" : { "name":"John", "age":30, "city":"New York" }

}

* một mảng

{

"employees":[ "John", "Anna", "Peter" ]

}

* một Boolean

{ "sale" : true }

* null

{ "middlename":null }

Giá trị JSON không được là một trong các loại dữ liệu sau:

* một hàm
* một ngày (thời gian)
* undefined
  1. **Đối tượng trong JSON**
* Cú pháp:

VD: { "name":"John", "age":30, "car":null }

* Truy cập đối tượng:

VD: myObj = {"name" : "John", "age" : 30, "car" : null};

x = myObj.name; // hoặc x = myObj["name"];

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

* Vòng lặp một đối tượng

VD: var myObj = {"name" : "John", "age" : 30, "car" : null};

for (x in myObj) {

document.getElementById("demo").innerHTML += x + "<br>";

}

Trong vòng lặp for-in, sử dụng ký hiệu ngoặc vuông để truy cập các giá trị thuộc tính :

VD: var myObj = {"name" : "John", "age" : 30, "car" : null};

for (x in myObj) {

document.getElementById("demo").innerHTML += myObj[x] + "<br>";

}

Các đối tượng lồng nhau:

VD: myObj = {  
     "name":"John",  
     "age":30,  
     "cars": {  
       "car1":"Ford",  
         "car2":"BMW",  
         "car3":"Fiat"  
     }  
  }

Truy cập đối tượng lồng nhau:

document.getElementById("demo").innerHTML += myObj.cars.car2 ;

//or:

document.getElementById("demo").innerHTML += myObj.cars["car2"];

Sửa đổi giá trị

myObj.cars.car2 = "Mercedes";

document.getElementById("demo").innerHTML =myObj.cars.car2;

//or:

myObj.cars["car2"] = "Mercedes";

document.getElementById("demo").innerHTML = myObj.cars["car2"];

Xóa thuộc tính:

delete myObj.cars.car2;

* 1. **Chuyển các dữ liệu thành chuỗi**
* Chuyển đối tượng thành 1 chuỗi:

document.getElementById("demo").innerHTML = JSON.stringify(obj);

* Chuyển mảng thành 1 chuỗi:

document.getElementById("demo").innerHTML = JSON.stringify(arr);

* 1. **Hàm JSON.parse()**

Hàm JSON.parse dùng để chuyển đổi văn bản thành một đối tượng JavaScript:

VD: var txt = '{"name":"John", "age":30, "city":"New York"}'

var obj = JSON.parse(txt);

document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;

* 1. **Mảng trong JSON**

Mảng là đối tượng JSON

Trong JSON, giá trị mảng phải là kiểu chuỗi, số, đối tượng, mảng, boolean hoặc null. Trong JavaScript, các giá trị mảng có thể là tất cả các điều trên, cộng với biểu thức JavaScript hợp lệ khác, bao gồm các hàm, ngày tháng và undefined.

VD: {

"name":"John",

"age":30,

"cars":[ "Ford", "BMW", "Fiat" ]

}

* Truy cập các giá trị mảng

VD: var myObj;

myObj = {

"name":"John",

"age":30,

"cars":[ "Ford", "BMW", "Fiat" ]

};

document.getElementById("demo").innerHTML = myObj.cars[2];

* Vòng lặp qua 1 mảng:

VD: for (i in myObj.cars) {

x += myObj.cars[i] + "<br>";

}

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

Mảng lồng nhau trong đối tượng JSON: Giá trị trong một mảng cũng có thể là mảng khác, hoặc thậm chí là một đối tượng JSON khác.

VD: myObj = {  
    "name":"John",  
    "age":30,  
    "cars": [  
        { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },  
        { "name":"BMW", "models":[ "320", "X3", "X5" ] },  
        { "name":"Fiat", "models":[ "500", "Panda" ] }  
    ]  
 }

Để truy cập các mảng bên trong mảng, sử dụng vòng lặp for-in cho mỗi mảng:

VD: var myObj, i, j, x = "";

for (i in myObj.cars) {

x += "<h2>" + myObj.cars[i].name + "</h2>";

for (j in myObj.cars[i].models) {

x += myObj.cars[i].models[j] + "<br>";

}

}

* Sửa đổi giá trị trong mảng:

VD: myObj.cars[1] = "Mercedes";

* Xóa mục trong mảng:

VD: delete myObj.cars[1];

1. **Local Strorage**

Local Stroragecho phép lưu trữ thông tin trên web tương đối lớn lên đến 5MB

Có 2 loại Local Storage:

* window.localStorage: Lưu trữ dữ liệu vô thời hạn, dữ liệu sẽ được lưu trữ cho tới khi người dùng clear history.
* window.sessionStorage: Lưu trữ dữ liệu cho mội phiên làm việc, có nghĩa dữ liệu sẽ bị mất khi bạn tắt browser.

Kiểm tra xem trình duyệt có hỗ trợ Local Storage không bằng cách:

if (typeof(Storage) !== "undefined") {

// Code for localStorage/sessionStorage.

} else {

// Sorry! No Web Storage support..

}

* **localStrorage** (Lưu trữ cục bộ) : Lưu trữ dữ liệu vô thời hạn, dữ liệu sẽ được lưu trữ cho tới khi người dùng clear history.

**Khởi tạo localStorage:**

localStorage.setItem("key", "value");

// or

localStorage.key = "value";

**Sử dụng:**

localStorage.getItem("key");

// or

localStorage.key;

**Xoá:**

localStorage.removeItem("key");

* **sessionStorage** cũng tương tự như localStorage, chỉ có một điểm khác đó là dữ liệu của sessionStorage sẽ mất khi bạn đóng trình duyệt.