**Javascript and java are completely different, both in concept and design**

In HTML, JavaScript code must be inserted between <script> and </script> tags.

A JavaScript function is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

**What is JavaScript (JS)? Why and When to use it?**

* JS is programming language of html and web
* Ví dụ : gắn sự kiện đúp chuột cho 1 phần tử nào đó
* Nên làm thành 1 file .js để dễ thao tác

**How many way to add JS into site?**

* **Dùng trực tiếp :** bằng cách sử dụng tag **<script></script> giữa tag <head> hoặc <body> ( thường dùng trong thẻ <head> hơn )**
* External : dùng thẻ .js như .css
* **Câu lệnh :** <script src="myScript.js"></script>

**Understand JS syntax? The operator? Loop?**

* **Js syntax** : cú pháp của JS, tổng hợp lại các quy tắc mà chương trình xây dựng
  + Ví dụ :
  + var x, y, z;        // Khai báo biến  
    x = 5; y = 6;      // Gán giá trị  
    z = x + y;         // Tính giá trị
  + **The JavaScript syntax defines two types of values**: Fixed values and variable values. ( giá trị cố định và giá trị biến )
    - **Fixed values**( cố đinh) : các số được viết có hoặc ko số thập phân
      * **Number** : 10.05 , 1001
      * **String**(Chuỗi là văn bản được viết trong dấu ngoặc kép hoặc đơn ) : “Hung” , ‘hung’
    - variable values : keyword “ var”
      * ví dụ :
      * var x;  
        x = 6;
* **The operator : sử dụng toán tử số học ( + - \* / ) để tính toán giá trị**
  + **Ví dụ 1 :**
  + <script>
  + document.getElementById("demo").innerHTML = 5 \* 10;
  + </script>
  + **Ví dụ 2 :**
  + <script>
  + var x;
  + x = 5;
  + document.getElementById("demo").innerHTML = x \* 10;
  + </script>
  + **Ví dụ 3 :**
  + <script>
  + document.getElementById("demo").innerHTML = "John" + " " + "Doe";
  + </script>
* **Loop : Vòng lặp** 
  + if you want to run the same code over and over again, each time with a different value.
  + **Ví dụ** :
    - Thay vì :
    - text += cars[0] + "<br>";   
      text += cars[1] + "<br>";   
      text += cars[2] + "<br>";   
      text += cars[3] + "<br>";   
      text += cars[4] + "<br>";   
      text += cars[5] + "<br>";
    - Vòng lặp :
    - var i;  
      for (i = 0; i < cars.length; i++) {   
        text += cars[i] + "<br>";  
      }
  + **Các loại vòng lặp** :
    - **For** : lặp thông qua code nhiều lần
    - **For/in** : lặp thông qua các thuộc tính của một đối tượng
    - **For/of** : lặp thông qua các giá trị của một đối tượng lặp
    - **While** : lặp thông qua code khi một điều kiện dc set true
      * **Ví dụ** :
      * while (i < 10) {  
          text += "The number is " + i;  
          i++;  
        }
    - **Do/while** : thực thi trước mã khối 1 lần trước khi kiểm tra xem dk có đúng ko, sau đó nó sẽ lặp lại vòng lặp miễn là đúng ( nếu ko đúng ddkien trong vòng while thì chỉ in ra biến trong “do”
      * **Ví dụ** : <script>
      * var text = ""
      * var i = 0;
      * do {
      * text += "<br>The number is " + i;
      * i++;
      * }
      * while (i < 10);
      * document.getElementById("demo").innerHTML = text;
      * </script>
    - **ví dụ :**
    - for (*statement 1*;*statement 2*;*statement 3*) {  
        // *code block to be executed*  
      }
    - <script>
    - var text = "";
    - var i;
    - for (i = 0; i < 5; i++) {
    - text += "The number is " + i + "<br>";
    - }
    - document.getElementById("demo").innerHTML = text;
    - </script>
    - **Kết quả :**
    - The number is 0  
      The number is 1  
      The number is 2  
      The number is 3  
      The number is 4
    - *statement 1 thực thi 1 lần trước khi câu lệnh khối mã thực thi*
    - *statement 2 định danh điều kiện cho mã khối*
    - *statement 3 thực thi ( mọi lúc) sau khi mã khối dc thực thi*
    - *so sánh while và for*
      * *for bắt buộc phải có 3 satement*
      * *while chỉ cần có statement điều kiện là dc*
      * var cars = ["BMW", "Volvo", "Saab", "Ford"];  
        var i = 0;  
        var text = "";
      * for (;cars[i];) {  
          text += cars[i] + "<br>";  
          i++;  
        }
      * hoặc
      * while (cars[i]) {  
          text += cars[i] + "<br>";  
          i++;  
        }

**How many data type in JS?**

**Kieur dữ liệu**

* **String** : cho chuoix dạng chữ
* **Number** : dạng số decimals hoặc ko decimals
* **Boolean** : sẽ chỉ có 2 giá trị true hoặc false
* Undefined : 1 biến ko có giá trị sẽ cho giá trị undefined
* Null : có nghĩa là giá trị rỗng hoặc giá trị không tồn tại, nó có thể được sử dụng để gán cho một biến như là một đại diện không có giá trị.
* 3 kiểu đối tượng chính ( object, array, regex)
* **Ví** **dụ** :
* <script>
* var x = 5;
* var y = 5;
* var z = 6;
* document.getElementById("demo").innerHTML =
* (x == y) + "<br>" + (x == z);
* </script>
* **Kết** **quả** :
* true  
  false
* **Arrays**( **mảng**) : được phân tách bằng dấu phảy, bên trong là các giá trị
* **Ví** **dụ** : tạo mảng cars
* var cars = ["Saab", "Volvo", "BMW"];
* JS objects ( đối tượng js) : viết bằng {}
* Ví dụ :
* <script>
* var person = {
* firstName : "John",
* lastName : "Doe",
* age : 50,
* eyeColor : "blue"
* };
* document.getElementById("demo").innerHTML =
* person.firstName + " is " + person.age + " years old.";
* </script>
* **Kết quả :**
* John is 50 years old.
* =>
* typeof "John"              // Returns "string"   
  typeof 3.14                // Returns "number"  
  typeof true                // Returns "boolean"  
  typeof false               // Returns "boolean"  
  typeof x                   // Returns "undefined" (if x has no value)

**Different between undefined and null ?**

* typeof undefined           // undefined  
  typeof null                // object  
    
  null === undefined         // false  
  null == undefined          // true
* equal in value but different in type

**How to convert data type of variable?( chuyển đổi loại dữ liệu của biến)**

**Hiểu ngầm : được gọi là ép kiểu dữ liệu cho 1 biến, vì JS nhận nhiều giá trị ko mog muốn nên sẽ luôn cố gắng chuyển đổi các giá trị thay vì từ chối nó.**

**Hiểu chi tiết : Các biến JS có thẻ chuyển đổi thành 1 biến mới hoặc 1 dạng dữ liệu khác**

* **chuyển từ numbers to string : số sang chuỗi, có thể sử dụng với các loại số, chữ , biến, biểu thức . VD:**
* **cú pháp String()**
* <https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_string>
* **Chuyển từ booleans to string : kiểu booleans sang chuỗi :** String(false)      // returns "false"
* String(true)       // returns "true"
* **Chuyển từ Dates sang Strings : từ kiểu ngày sang chuỗi :**
* String(Date())  // returns "Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)"
* Và 1 số loại method khác để chuyển dates to strings
* **Chuyển từ Strings to numbers :**
* **Ví dụ :** String chứa number (like “3.14”) convert to number (like 3.14) bằng cách Number("3.14")
* **Toán tử unary+ chuyển 1 biến sang 1 số :**
* Ví dụ : <https://www.w3schools.com/js/tryit.asp?filename=tryjs_typeof4>
* **Chuyển từ Booleans to numbers :**
* Number(false)     // returns 0  
  Number(true)      // returns 1
* **Chuyển từ Dates to number :**
* d = new Date();  
  Number(d)          // returns 14045680277
* **bảng chuyển đổi kiểu dữ liệu:** <https://www.w3schools.com/js/js_type_conversion.asp>

**How to define and call function in JS?**

**Gọi phương thức của js : là khối mã dc thiết kế để thực hiện nvu cụ thẻ**

* **một function là 1 khối mã lệnh sẽ được thực thi khi dc gọi tới. nếu có 1 vài dòng code cần sử dụng nhiều lần, tạo 1 function bao gồm các code muốn lặp lại và sau này chỉ cần gọi tên cái function đấy ra**

**tạo function : dùng từ khoá function functionname(){ code thực thi}**.

* **Ví dụ : gọi 1 function để perform 1 phép tính và trả về kết quả**
* **<script>**
* function myFunction(p1, p2) {
* return p1 \* p2;
* }
* document.getElementById("demo").innerHTML = myFunction(4, 3);
* **</script>**
* **🡪 kết quả : 12**
* **Cấu trúc :**
* function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
  }

**Local and Global variable in JS? How to use it?**

**Local: biến cục bộ khai báo bên trong 1 khối lệnh và chỉ thực hiện bên trong khối lệnh**

**Ví dụ :** // code here can NOT use carName  
  
function myFunction() {  
  var carName = "Vox`lvo";  
  
  // code here CAN use carName  
  
}

Global : biến toàn cục : được khai báo bên ngoài khối lệnh, có thể htuwjc hiện bất cứ đâu.

Example

var carName = "Volvo";  
  
// code here can use carName  
  
function myFunction() {  
  
  // code here can also use carName   
  
}

**toàn bộ lệnh và chức năng đều có thể truy cập vào biến này**

**Build-in Object ( xây dựng đối tượng )**

**Dates ( ngày)**

**Có 4 cách tạo 1 đối tượng date mới**

* **new Date() :** tạo đối tượng date với thời gian hiện tại
* ví dụ : <https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new>  
  - **new Date(year, month, day, hours, minutes, seconds, milliseconds)** : tạo đối tượng date với thời gian quy định  
  - **new Date(milliseconds) :** tạo một đối tượng date như thời gian cộng 0 + mili giây (JS lưu trữ ngày tháng bằng số một phần nghìn giây từ January 01, 1970, 00:00:00 UTC (Universal Time Coordinated).
* Ví dụ : <https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_zero>  
  - **new Date(date string)** : tạo 1 đối tượng date từ 1 chuỗi date
* Ví dụ : <https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_string>

**Array ( mảng) : sử dụng đẻ lưu trữ nhiều giá trị trong 1 biến :**

**Ví dụ :** var cars = ["Saab", "Volvo", "BMW"];

**String : ( chuỗi ) sử dụng để lưu trữ và thao tác văn bản**

**Chuỗi js là 0 hoặc nhiều kí tự dc viết bên trong dấu ngoặc kép**

var x = "John Doe";

**Number : js chỉ có 1 dạng số , số có thể viết dưới dạng số thập phân hoặc ko**

var x = 3.14;    // A number with decimals  
var y = 3;       // A number without decimals

**Is Javascript an OOP?**

**Đối tượng là 1 thực thể có trạng thái và hạnh vi, 1 đối tượng là 1 trường hợp của 1 lớp**

Javascript là một ngôn ngữ hỗ trợ cả **OOP**(hướng đối tượng) và **Functional Language**(ngôn ngữ chức năng)

Bốn tính chất của hướng đối tượng:  
– Tính đóng gói(**encapsulation**)  
– Tính trừu tượng(**abstraction**)  
– Tính đa hình(**polymorphism**)  
– Tính thừa kế(**inheritance**)

Xoay quanh các đối tượng, Tạo ra những class , các đối tượng kế thừa từ class và bên trong đối tượng sẽ là thuộc tính giá trị, hàm , ……..

In JavaScript, almost "everything" is an object.

* **Booleans** can be objects (if defined with the new keyword)
* **Numbers** can be objects (if defined with the new keyword)
* **Strings** can be objects (if defined with the new keyword)
* **Dates** are always objects
* **Maths** are always objects
* **Regular** **expressions** are always objects
* **Arrays** are always objects
* **Functions** are always objects
* **Objects** are always objects

All JavaScript values, except primitives, are objects.

**How to create new object in JS?**

* **object literal.( đối tượng bằng chữ)**
  + <script>
  + var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
  + document.getElementById("demo").innerHTML =
  + person.firstName + " is " + person.age + " years old.";
  + </script>
  + 🡪 kết quả : John is 50 years old.
* **keyword new.**
  + **<script>**
  + var person = new Object();
  + person.firstName = "John";
  + person.lastName = "Doe";
  + person.age = 50;
  + person.eyeColor = "blue";
  + document.getElementById("demo").innerHTML =
  + person.firstName + " is " + person.age + " years old.";
  + **</script>**
* **Nên dùng cách đầu tiên**

**How to access object property? Đối tượng sở hữu**

* *objectName.propertyName*
* *ví dụ :*
* **var person = {**
* **firstName: "John",**
* **lastName : "Doe",**
* **id : 5566**
* **};**
* **// Display some data from the object:**
* **document.getElementById("demo").innerHTML =**
* **person.firstName + " " + person.lastName;**
* **🡪 KẾT QUẢ :** John Doe
* *objectName["propertyName"]*

**How to call method of object? ( gọi phương thức của một dối tượng)**

**Phương thức call() là phương thức được xác định trước , được sử dụng để gọi một phương thức với dối tượng chủ sở hữu như một tham số**

**Ví dụ : gọi phương thức fullName của person, sử dụng nó trên person1**

var person = {  
  **fullName**: function() {  
    return this.firstName + " " + this.lastName;  
  }  
}  
var person1 = {  
  firstName:"John",  
  lastName: "Doe"  
}  
var person2 = {  
  firstName:"Mary",  
  lastName: "Doe"  
}  
person.fullName.call(**person1**);

**kết quả trà về : John Doe**

**What is object prototype? When prototype is used?**

* **Prototype : thông thường tạo đối tượng thì phải có class, còn prototype thì ko cần class mà sẽ có thể tạo đối tượng luôn từ một nguyên mẫu**
* **Tạo đối tượng từ 1 đối tượng khác**
* Tất cả các đối tượng JavaScript kế thừa các thuộc tính và phương thức từ một nguyên mẫu.(prototype)
* Date objects inherit from Date.prototype
* Array objects inherit from Array.prototype
* Person objects inherit from Person.prototype

The Object.prototype is on the top of the prototype inheritance chain:

Date objects, Array objects, and Person objects inherit from Object.prototype

* Nguyên mẫu.(**prototype**) Khi một chức năng được tạo trong JavaScript, công cụ JavaScript sẽ thêm một thuộc tính nguyên mẫu vào chức năng. Thuộc tính nguyên mẫu này là một đối tượng (được gọi là đối tượng nguyên mẫu) có thuộc tính constructor theo mặc định. thuộc tính constructor trỏ lại chức năng mà đối tượng nguyên mẫu là thuộc tính
* **Dùng khi :**
* **The JavaScript prototype property allows you to add new properties to object constructors: thêm 1 thuộc tính vào đối tượng mà ko cần phải trong function**
* **<script>**
* function Person(first, last, age, eye) {
* this.firstName = first;
* this.lastName = last;
* this.age = age;
* this.eyeColor = eye;
* }
* Person.prototype.nationality = "English";
* var myFather = new Person("John", "Doe", 50, "blue");
* document.getElementById("demo").innerHTML =
* "The nationality of my father is " + myFather.nationality;
* **</script>**
* **🡪 KẾT QUẢ :** The nationality of my father is English
* **The JavaScript prototype property also allows you to add new methods to objects constructors: thêm 1 phương thức vào obj**
* **<script>**
* function Person(first, last, age, eye) {
* this.firstName = first;
* this.lastName = last;
* this.age = age;
* this.eyeColor = eye;
* }
* Person.prototype.name = function() {
* return this.firstName + " " + this.lastName
* };
* var myFather = new Person("John", "Doe", 50, "blue");
* document.getElementById("demo").innerHTML =
* "My father is " + myFather.name();
* **</script>**
* **🡪 kết quả :** My father is John Doe

**What is regex? When to use it?**

* Ví dụ như bạn có thể check chuỗi bao gồm chữ hoặc số, số lượng kí tự, vị trí của kí tự, chữ hoa, chữ thường, và nhiều hơn nữa.
* **Là một chuỗi các ký tự tạo thành một mẫu tìm kiếm, ktra, thay thế.**
* Ví dụ :
* **Cú pháp :** /pattern/modifiers; ( mẫu / bổ nghĩa)
* **Ví dụ :** var patt = /w3schools/i;
* **/w3schools/i**  is a REGEX
* **w3schools**  is a pattern (to be used in a search).
* **i**  is a modifier ( sửa đổi tìm kiếm ko phân biệt chữ hoa hay thường
* In JavaScript, regular expressions are often used with the two **string methods**: search() and replace().
* **Search : phương thức tìm kiếm 1 string cho giá trị dc chỉ định và trả về vị trí được tìm thấy . VD:**
* var n = str.search(/w3Schools/i);
* <https://www.w3schools.com/js/tryit.asp?filename=tryjs_regexp_string_search>
* **Repalce : trả về 1 string sửa đổi nơi mà pattern được thay thế :**
* **var txt = str.replace(/microsoft/i,"W3Schools"); // thay microsoft bằng w3s…**
* <https://www.w3schools.com/js/tryit.asp?filename=tryjs_regexp_string_replace>
* **Bracket( ngoặc vuông) : dùng để tìm 1 loạt các ký tự trong phạm vi**
* **Metacharacters** : là những ký tự với ý nghĩa đặc biệt như là tìm số \d, tìm khoảng trắng \s
* **Quantifiers** : xác định số lượng , ví dụ tìm string có chứa ít nhất 1 chữ n : n+ hay tìm string chứa từ 0 hoặc nhiều hơn n lần xuất hiện n\*
* **Modifiers** : là công cụ sửa đổi , thực hiện tìm kiếm toàn cục ko phân biệt chữ hoa hay thường \i , \g, \m

**JSON :**  **J**ava**S**cript **O**bject **N**otation. ( ký hiệu đối tương)

Là một chuỗi có cấu trúc :

var myJSON = '{"name":"John", "age":31, "city":"New York"}';

* Use the **JavaScript** function **JSON**.stringify() to **convert** it into a **string**.
  + **<script>**
  + **var obj = { name: "John", age: 30, city: "New York" };**
  + **var myJSON = JSON.stringify(obj);**
  + **document.getElementById("demo").innerHTML = myJSON;**
  + **</script>**
  + **🡪 kết quả :** {"name":"John","age":30,"city":"New York"}

**How to parse JSON String into js object?**

* **Khi nhận dữ liện từ web server thì dữ liệu sẽ là dạng chuỗi string**
* **Phân tích (parse) dữ liệu với** JSON.parse() và dữ liệu sẽ trở thành đối tượng js
* **Ví dụ :**
* **Nhận dữ liệu từ trang web đang có dạng string**
* '{ "name":"John", "age":30, "city":"New York"}'
* Sử dụng JSON.parse() để chuyển về object :
* var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
* cụ thẻ :
* **<script>**
* **var txt = '{"name":"John", "age":30, "city":"New York"}'**
* **var obj = JSON.parse(txt);**
* **document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;**
* **</script>**
* **🡪 kết quả : John,30**

**Dom and EVENT processing**

**What is Dom ? document object model : mô hình đối tượng tài liệu**

**Dùng để truy cập và thay đổi tất cả các yếu tố của tài liệu html**

**Khi trang web load, sẽ tự tạo ra 1 DOM**

**Có thể thay đổi, cập nhật, thêm mới hoặc gỡ bỏ nội dung, cấu trúc của các phần tử, thuộc tính, style**

**how many way to find element in DOM? ( tìm phần tử)**

**Tìm phần tử theo ID:** <https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementbyid>

**Tìm theo Tag name :**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementsbytagname2>

**Tìm theo Class Name : tìm tất cả các phần tử cùng class name**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementsbyclassname>

**tìm theo CSS selector ví dụ p.intro**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_queryselectorall>

**tìm theo HTML collection object ( tập hợp đối tượng) sẽ in ra toàn bộ text trong phần value**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_form_elements>

**how to find child element ?**

**Thuộc tính con trả về 1 tập hợp các phần tử con**

**Ví dụ :var c = document.body.children;**

<https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_element_children>

Cú pháp element.children

**How to loop child elements ?**

**How to change content/atributes of elements? Ví dụ đổi màu, background của 1 phần tử:**

**Dùng cú pháp :** document.getElementById(*id*).style.*property*=*new style*

*document.getElementById("p2").style.color = "blue";*

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_change_style>

sử dụng bằng event onclick

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_color2>

**How to remove or insert element in DOM ?**

Thêm mới 1 phần tử DOM :

Tạo div, p1 p2 in ra màn hình

Trong js tạo 1 obj A bằng cú pháp document.createElement(“p”);

Sau đó tạo 1 text cho nó B : document.createTextNode(“text”)

Nối biến B vào A : A.appendChild(B);

Tạo obj cho thẻ div là C

Nối A vào C : C.appendChild(A);

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_elementcreate>

**Dùng insertBefore để dưa phần tử được appendchild lên đầu**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_elementcreate2>

**Remove element**

**Tạo 1 thẻ div, p**

**Sau đó dùng JS để tạo ra đối tượng tìm tới thẻ div và p ( getElement….)**

**Đặt tên obj cho div là a, obj cho p là b**

**a.removeChild(b);**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_elementremove>

**replace eleemtn**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_elementreplace>

Dom event LISTENER

**What is event ? là 1 sự kiện thực thi cho 1 phần tử ví dụ : khi người dùng click vào 1 phần tử html thì sẽ có sự kiện gì xảy ra**

**Khi click vào chuột**

**Khi webpage load**

**Khi ảnh load**

**Khi chuột di chuyển qua phần tử**

**Khi người dùng vuốt 1 phím**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick2>

**Listener ?**

**Phương thức addEventListener() đính kèm 1 trình xử lý sự kiện vào phần tử chỉ định và nó sẽ ko bị ghi đè vào các trình xử lý sự kiện hiện có**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_displaydate>

**có thể cho nhiều event xử lý vào 1 phần tử**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add_many2>

**có thể cho nhiều event xử lý vào của cùng 1 loại vào 1 phần tử, ví dụ như có 2 “click” event trong cùng 1 phần tử**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add_many>

**sử dụng phương thức** removeEventListener()  để gỡ 1 sự kiện

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_remove>

Nếu một phần tử DOM bị loại bỏ không có tham chiếu (không có tham chiếu trỏ đến nó) thì có - phần tử được chọn bởi trình thu gom rác cũng như bất kỳ trình xử lý / trình lắng nghe sự kiện nào liên quan đến nó.

Nghĩa là :

var a = document.createElement('div');

var b = document.createElement('p');

// Add event listeners to b etc...

a.appendChild(b);

a.removeChild(b);

b = null;

// A reference to 'b' no longer exists

// Therefore the element and any event listeners attached to it are removed.

Như này là sẽ mất cả listener do b=null ( rỗng ) – tham chiếu rỗng

Còn như này là vẫn còn do b vẫn còn tham chieus là “p”

|  |  |
| --- | --- |
|  | var a = document.createElement('div'); |
|  | var b = document.createElement('p'); |
|  | // Add event listeners to b etc... |
|  | a.appendChild(b); |
|  | a.removeChild(b); |
|  | // A reference to 'b' still exists |
|  | // Therefore the element and any associated event listeners are still retained. |

**Có 2 cách truyền sự kiện là bubblling và capturing : khi một sự kiện dc truyền vào phần tử thì sẽ có 2 cách truyền để tới dc sự kiện đó**

**Cý : event focus không có cơ chế**

**Bubbling :** 

<style>

body \* {

margin: 10px;

border: 1px solid blue;

}

</style>

<form onclick="alert('form')">FORM

<div onclick="alert('div')">DIV

<p onclick="alert('p')">P</p>

</div>

</form>

**Giải thích : thẻ p kích hoạt event và arlert ra text**

**Tiếp tục bubbling lên thẻ div và arlert ra text div**

**Tiếp tục bubbling thẻ form**

**Các phần tử con sẽ “up through” đến phần tử cha( nổi bọt nước )**

**Capturing :** Theo tiêu chuẩn [Dom Events](http://www.w3.org/TR/DOM-Level-3-Events/) mô tả rất rõ 3 giai đoạn điều hướng của event

1. Capturing - Trigger event lần lượt từ DOM parent đến DOM đã đăng kí event
2. Target - Đây là phần từ DOM đăng kí event
3. Bubbling - ngược lại với capturing, sẽ trigger từ DOM đăng kí event bubbling đến DOM parent

**Capturing thì nuguwcoj lại sẽ thực hiện event lần lượt event từ thẻ cha ngoài cùng cho tới thẻ muốn trỏ tới**

**Event cho input :**

**Onblur : khi người dùng rời khỏi ô input thì sự kiên sẽ diễn ra ( ví dụ gõ text là dấu thường nhưng khi ấn ra ngoài sẽ thì text sẽ thành viết hoa )**

<https://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onblur>

onchange : ???? giống hết blur

**Onfocus** : khi 1 input field dc ấn vào

**Onsubmit : khi người dùng click vào nút**

**Onselect**

**Onreset**

**Onkeydown**

**Onkeypress**

**onkeyup**