**Javascript**

Javascript\_01

1. Sử dụng Javascript trong HTML

+ Nhúng mã script vào html

**<script language=“javascript”> ... </script>**

+ Xuất nội dung bằng Js

document.write(“Nội dung muốn hiển thị”);

+ Comment trong Js: /\*…\*/

+ Nhúng file .js từ ngoài vào:

<script type=“text/javascript” language=“javascript” src=“URL”></script>

1. Giao tiếp với người dùng:

+ alert(“Nội dung muốn thông báo”);

+ Prompt(“Nội dung tương tác”);

1. Khai báo biến trong js: var ten\_bien;
2. Các kiểu dữ liệu trong Js

+ ineterger/ float/ double/ string/ boolean/ array/ object.

* Primitives
  + Number
  + String
  + Boolean
* Special types
  + Undefined
  + Null
* Reference types
  + Object: mô tả vật thể có nhiều thuộc tính khác nhau

Var objectName = {

Key: value

};

Truy xuất vào key:

Cách 1: objectName.key

Cách 2: objectName[‘key’]

* + Array

Var arrayName = [1, 4, 5];

1. Toán tử trong Js

+ Toán tử gán: “=”

+ Toán tử nối: “.” : bien = “chuoi”.bien2;

+ Toán tử số học

+ Toán tử so sánh

+ Toán tử logic

+ Toán tử kết hợp

Javascript\_02

1. Biểu thức điều kiện

+ If/ if\_else/ if\_else if\_else

+ Switch\_case\_default ( note: break;)

1. Vòng lặp

+ While

+ Do \_ While

+ For(…;…;…)

Javascript\_03

1. **Đống tượng mảng**

+ Khai báo mảng:

**Var tenMang = new Array();**

**Var tenMang = new Array(value1, value2, … valueN);**

+ Thêm một phần tử vào mảng:

**tenMang[Key] = Value;**

+ Xuất giá trị phần tử trong mảng:

**Document.write(tenMang[Key]);**

+ Duyệt mảng

**for(x in tenMang){**

**document.write(tenMang[x]);**

**}**

1. **Một số phương thức làm việc với mảng**

+ Phương thức Length: **nameArray.length;** (Không phải hàm “()”)

+ Phương thức concat() dùng để gom tất cả các phần tử của nhiều mảng riêng rẽ tập trung vào một mảng, trả về trong mảng mới : **nameArray.concat();**

( Mang = mang1.concat(mang2) )

+ Phương thức join() giúp chúng ta tạo ra một chuỗi từ các phần tử trong một mảng và được ngăn cách bằng một ký tự bấy kỳ : **mang.join();**

+ Phương thức pop() dùng để xóa phần tử cuối cùng trong một mảng, trả về phần tử bị đẩy ra khỏi mảng: **mang.pop();**

+ Phương thức push() dùng để thêm một hoặc nhiều phần tử vào cuối mảng, trả về độ dài mảng mới:

**mang.push();**

+ **array.shift() :** đẩy phần tử đầu tiên ra khỏi mảng, trả về phần tử bị đẩy ra.

+ **array.unshift() :** thêm phần tử vào đầu mảng, trả về độ dài mới của mảng.

+ Phương thức splice() sử dụng để kết hợp việc xóa phần tử cũ khỏi mảng đồng thời thêm một phần tử mới tiếp sau các phần tử còn lại trong mảng:

**mang.splice(index, count, element1, …elementN);**

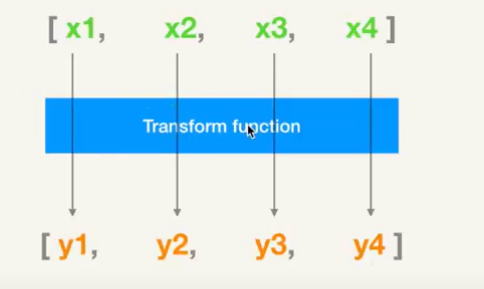
**Trong đó**

* **index**: vị trí phần tử muốn loại
* **count**: Số phần tử muốn loại tính từ vị trí index
* **element1**,…element là các phần tử mới thêm vào

+ Sắp xếp mảng theo theo thứ tự tăng dần : **mang.sort() ;**

+ Phương thức reverse() trả lại một mảng với thứ tự các phần tử bị đảo ngược: **mang.reverse();**

* **Array map:** ánh xạ, biến đổi các phần tử cũ thành các phần tử mới.



Syntax:

**array.map(function(item){**

**//transform**

**return newValue;**

**});**

* **Array filter:** lọc các phần tử.

Syntax:

**array.filter(function(item){**

**//filter**

**return newValue;**

**});**

Ví dụ: lọc các số lẻ của mảng

var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

var eventNumbers = numbers.filter(function(item){

return item%2 ===1;

});

console.log(eventNumbers);

//-> 1, 3, 5, 7, 9

* **Array find:** tìm kiếm phần tử.

Tương tự với **array.fillter**, khi tìm thấy phần tử thỏa mãn sẽ dừng lại.

Syntax:

**array.find(function(item){**

**//find**

**return newValue;**

**});**

Ví dụ: lọc các số lẻ của mảng

var numbers = [2, 2, 6, 4, 5, 6, 7, 8, 9, 10];

var eventNumbers = numbers.find(function(item){

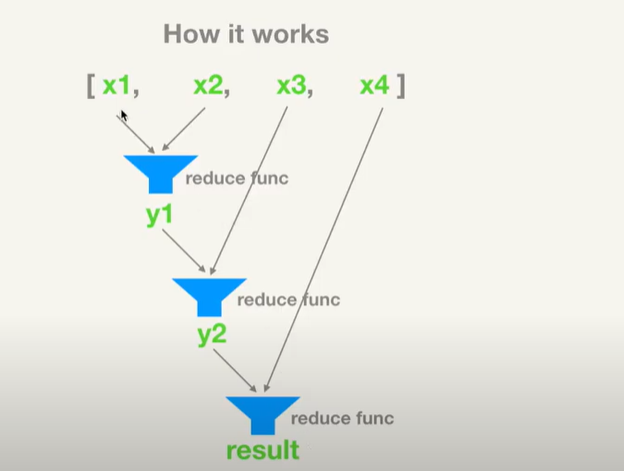
return item%2 ===1;

});

console.log(eventNumbers);

//-> 5

* **Array reduce:** Phương thức reduce() dùng để thực thi một hàm lên từng phần tử của mảng (từ trái sang phải) với một biến tích lũy để thu về một giá trị duy nhất



Sytax:

**array.reduce(function(item1, item2){**

**//caculation**

**return newValue;**

**});**

Ví dụ: tính tổng

//array reduce: hàm giảm

var numbers = [1, 2, 4, 7, 5, 12, 23];

var numReduce = numbers.reduce(function(red1, red2){

return red1 + red2;

});

console.log(numReduce);

//->54

Syntax 2:

**array.reduce(function(item1, item2){**

**//caculation**

**return newValue;**

**}, init)** *//init: giá trị khởi tạo ban đầu, thay cho item1*

3. Hàm trong Js

**function tenHam( x1, x2,…){**

**/\* Các lệnh thực thi Hàm \*/**

**return ketQua;**

**}**

**+** Object methods ( các phương thức của object)

var myDog = {

weight: 8,

name: 'Bull dog',

bark: function() { //anonymous function

console.log('Gâu gâu...')

}

};

console.log(myDog.bark());

+ Biến cục bộ/ biến toàn cục

1. Hàm xây dựng sẵn trong Js

+ Hàm isNaN() kiểm tra giá trị. Nếu không phải số trả về true, ngược lại thì nếu là số thì trả về là false

**Cú pháp**

**isNaN();**

+ Hàm eval() biến chuỗi thành biểu thức tính toán được hoặc biến chuỗi thành lệnh thi hành được như là mã lệnh của JS

**Cú pháp**

**Eval();**

+ Hàm SetTimeout() có tác dụng gọi một hàm sau thời gian định trước, và chỉ thực thi một lần duy nhất

**Cú pháp**

**setTimeout(“code”, Delaytime);**

**Trong đó**

* **code**: có thể là một hàm hoặc một câu lênh
* **Delaytime**: thời gian truyền vào được tính theo mily giây

+ Hàm SetInterval () có tác dụng gọi một hàm sau thời gian định trước, được thực thi trong suốt quá trình chạy file hiện hành

**Cú pháp**

**SetInterval (“code”, Delaytime);**

**Trong đó**

* **code**: có thể là một hàm hoặc một câu lênh
* **Delaytime**: thời gian truyền vào được tính theo mily giây

5. Hàm toán học trong Js

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Phương Thức | Chức Năng | Ví Dụ |
| 1 | Math.round(x) | Làm tròn | Math.round(5.6) = 6 |
| 2 | Math.floor(x) | Làm tròn xuống | Math.round(4.5) = 4 |
| 3 | Math.random() | Lấy số ngẫu nhiên 0 < x < 1 | Math.round(x) = 0.73… |
| 4 | Math.max(x,y,z,…N) | Lấy số có giá trị lớn nhất trong chuỗi số | Math.max(1,5,3,7) = 7 |
| 5 | Math.min(x,y,z,…N) | Lấy số có giá trị nhỏ nhất trong chuỗi số | Math.max(1,5,3,7) = 1 |
| 6 | Math.abs(x) | Trả về giá trị số dương | Math.abs(-9.5) = 9.5 |
| 7 | Math.PI | Hằng số PI | Math.PI = 3.14… |
| 8 | Math.sqrt(x) | Lấy căn bậc 2 của một số | Math.abs(9) = 3 |
|  |  |  |  |

6. Javascript callback

Dùng function như một biến, được truyền và gọi để thực hiện.

var coffeeMachine = {

makeCoffee: function (onFinish) {

console.log('Making coffee...');

onFinish();

}

}

var beep = function() {

console.log('Tít Tít...');

}

coffeeMachine.makeCoffee(beep);

console.log('==================================');

coffeeMachine.makeCoffee(function() {

console.log('Bíp Bíp...');

});

Javascript\_04: Sự kiện & đối tượng form

1. Sự kiện đối với các phần tử HTML

*Sự kiện Onclick mô tả hành động Click (Bấm) chuột vào bất cứ phần tử HTML nào*

**Cú pháp**

**onClick = “functionName();”**

*Sự kiện OnMouseOver mô tả hành động rê chuột lên bất cứ phần tử HTML nào.*

**Cú pháp**

**onMouseOver = “functionName();”**

*Sự kiện OnMouseOut mô tả hành động rê chuột ra khỏi một phần tử HTML bất kỳ.*

**Cú pháp**

**onMouseOut = “functionName();”**

1. Sự kiện đối với tài liệu HTML

*Sự kiện OnLoad mô tả trạng thái khi một trang web được tải lên*

**Cú pháp**

**onLoad = “functionName();”**

*Sự kiện onResize mô tả trạng thái khi thay đổi khung hiển thị trình duyệt web*

**Cú pháp**

**onResize = “functionName();”**

1. Sự kiện đối với một số phần tử form

*Sự kiện onSubmit mô tả trạng thái khi gửi dữ liệu trong Form qua nút Submit (Thuộc tính của phần tử Form)*

**Cú pháp**

**onSubmit = “functionName();”**

*Sự kiện onReset mô tả trạng thái khi làm mới dữ liệu trong Form qua nút Reset (Thuộc tính của phần tử Form)*

**Cú pháp**

**onReset = “functionName();”**

*Sự kiện onChange mô tả trạng thái khi thay đổi giá trị của một phần tử Form*

**Cú pháp**

**onChange = “functionName();”**

*Sự kiện onFocus mô tả trạng thái của một phần tử Form khi bị kịch hoạt*

**Cú pháp**

**onFocus = “functionName();”**

*Sự kiện onBlur mô tả trạng thái của một phần tử Form khi bị dời khỏi*

**Cú pháp**

**onBlur = “functionName();”**

1. Đối tượng form

*Đối tượng Form trong Javascript chính là phần tử Form trong HTML*

*Làm việc với Đối tượng Form chính là cách thức lấy giá trị từ các phần tử trong Form thông qua các phương thức của Đối tượng Form trong Javascript*

1. Lấy giá trị các phần tử form

*Làm việc với Đối tượng Form chính là cách thức lấy giá trị từ các phần tử trong Form thông qua các phương thức của Đối tượng Form trong Javascript*

Jquery\_01

1. Bắt đầu làm việc với jquery

+ Triệu gọi thư viên jquery: <script src=”url”></script>

+ Không gian làm việc với jquery: <script></script>

+ Cú pháp làm việc với jquery:

Cú pháp:

$(selector).action()

Trong đó:

 $: Từ khóa để bắt đầu một đối tượng mà jQuery sẽ làm việc

 Selector: HTML Tag, class, ID

 Action(): Một hoặc nhiều hành động của jQuery, nó chính là các phương thức (Hàm)

+ Chạy chương trình jquery:

$(document).ready(function(e) {

// vùng viết mã với jQuery

});

Trong đó:

 Document: đối tượng tài liệu HTM

 Ready(): Phương thức này sẽ tự động được thực thi mỗi khi trang được tải lên, tương tự như sụ kiện onLoad() trong Javascript

1. Jquery selector

+ HTML tag selector

Cú pháp:

$('HTML Tag name ')

+ ID selector

Cú pháp:

$('#ID name ')

+ Class selector

Cú pháp:

$(' .Class name')

1. Các sự kiện phổ biến trong jquery

+ Sự kiện click/ double click

Cú pháp:

$(‘selector’).click()/ $(‘selector’).dbclick

+ Sự kiện mouseenter

Cú pháp:

$(‘selector’). mouseenter ()

+ Sự liện mouseleave

Cú pháp:

$(‘selector’). mouseleave()

+ Sự kiện Mouse Up/Down

Cú pháp:

$(‘selector’).mousedown/up

(Sự kiện click/nhả click chuột trái hoặc phải hoặc giữa)

+ Sự kiện hover

Cú pháp:

$(‘selector’). hover (mouseenter (),mouseleave ())

* Sự kiện khi đưa chuột vào và đưa chuột ra

+ Sự kiện Focus/Blur/On

1. Jquery effects

+ Hiệu ứng Hide/Show

Cú pháp:

$(‘selector’).hide()

$(‘selector’).show()

* Có thể truyền tốc độ speed vào trong ( .hide(speed)).

$(‘selector’).toggle(speed, callback);

+ HIệu ứng Fade (Làm mờ)

Cú pháp:

$(‘selector’).fadeIn(speed)

$(‘selector’).fadeOut(speed)

$(‘selector’).fadeToggle(speed)

$(‘selector’).fadeTo()

($(selector).fadeTo(speed,opacity,callback);: cho phép làm mờ đến một độ đục opacity từ 0->1)

Trong đó:

 Speed: là tốc độ ẩn hiện được tính theo mily giây, các bạn cũng có thể để trống để sử dụng giá trị mặc định.

(FadeIn: làm mờ đi (cần display: none), FadeOut: làm rõ ra, FadeToggle = FadeIn/Out)

+ Hiệu ứng Slide (trượt )

Cú pháp:

slideDown(speed)

slideUp(speed)

slideToggle(speed)

Trong đó:

 Speed: là tốc độ ẩn hiện được tính theo mily giây, các bạn cũng có thể để trống để sử dụng giá trị mặc định.

+ Hiệu ứng Animate

Cú pháp:

$(‘selector’).animate({params}, speed)

+ Hiệu ứng Stop()

Cú pháp:

$(selector).stop(stopAll,goToEnd);

Phương thức jQuery stop () được sử dụng để dừng hoạt ảnh hoặc hiệu ứng trước khi nó kết thúc

+ Hiệu ứng Callback/Chaining (chạy nối tiếp (nối chuỗi) các câu lệnh Jquery

Trong đó

 Params: Đối tượng CSS

 Speed: Tốc độ thay đổi được tính theo mily giây hoặc bỏ qua để sử dụng giá trị mặc định

Jquery\_02

1. Jquery HTML

+ Các phương thức Get content

Cú pháp:

$(‘selector’).text()

$(‘selector’).html()

$(‘selector’).attr(‘attribute’)

$(‘selector’).val()

Trong đó:

 Text(): Chỉ lấy nội dung dạng text

 Html(): Lấy nội dung bao gồm cả text lẫn mã HTML

 Attr(): Lấy ra giá trị của thuộc tính truyền vào

 Val(): Lấy ra giá trị của thuộc tính Value (lấy từ ngoài đưa vào, <input/>)

+ Các phương thức Set content ( Thay đổi nội dung, gán giá trị)

Cú pháp:

$(‘selector’).text(‘text’)

$(‘selector’).html(‘string’)

$(‘selector’).attr({params})

$(‘selector’).val(value)

Trong đó:

 Text: Nội dung dạng Text

 String: Một chuỗi

 params: Đối tượng lưu trữ các thuộc tính và giá trị

 Value: Giá trị bất kỳ

+ Các phương thức thêm/xóa Class CSS trong jQuery

Cú pháp:

$(‘selector’).addClass(class1 class2 …)

$(‘selector’).removeClass(class1 class2 …)

$(‘selector’).toggleClass(class1 class2 …)

+ Các phương thức thêm thuộc tính CSS trong jQuery

Cú pháp:

$(‘selector’).css(‘property’, ‘value’)

Hoặc

$(‘selector’).css({‘property 1’:’ value 1’, ‘property 2’:’ value 2’,… ‘property N’:’value N’})

Trong đó:

 Property: Thuộc tính trong CSS

 Value: Giá trị tương ứng của từng thuộc tính cụ thể trong CSS

1. Jquery Traversing

+ Các phương thức lọc phần tử cha (Parent)

Cú pháp:

$(‘selector’).parent(‘element’)

$(‘selector’).parents(‘element’)

$(‘selector’).parentUntil(‘element’)

Trong đó:

 Parent(): Phương thức lọc ra tất cả các phần tử cha với cấp gần nhất. Nếu như có nhiều element khác nhau cùng cấp thì có thể lựa chọn chính xác một element nào đó.

 Parents(): Phương thức lọc ra tất cả các phần tử cha của tất cả các cấp. Nếu như có nhiều element khác nhau cùng cấp thì có thể lựa chọn chính xác một element nào đó.

 ParentsUntil(‘element’): Phương thức lọc ra tất cả các phần cha của tất cả các cấp và dừng lại trước element

+ Các phương thức lọc phần tử con (Children)

Cú pháp:

$(‘selector’). children()

$(‘selector’).find(‘element’)

$(‘selector’).find(‘\*’)

Trong đó:

 Children(): Phương thức lọc ra tất cả các phần tử con với cấp gần nhất. Nếu như có nhiều element khác nhau cùng cấp thì có thể lựa chọn chính xác một element nào đó.

 Find(): Phương thức lọc ra tất cả các phần tử con của một cấp nào đó theo element truyền vào.

 Find(‘\*’): Phương thức lọc ra tất cả các phần tử con của tất cả các cấp

+ Phương thức lọc phần tử bất kỳ (Eq)

Cú pháp:

$(‘selector’). eq(index)

Trong đó:

 Eq(): Phương thức lọc ra chính xác một Element theo vị trí index.

 Index: vị trí của Element trong một tập hợp Element được đánh số là những số nguyên dương bắt đầu từ 0 (tương tự như đánh số key của phần tử mảng)