# Java Script là gì

Java Script là một ngôn ngữ lập trình kịch bản, hiện đang sử dụng phổ biến để lập trình Web.

JavaScript được sử dụng rất phổ biến các trang web với mục đích phổ biến như sau:

* Kiểm tra thông tin người dùng nhập qua biểu mẫu
* Tạo hiệu ứng
* Thông báo
* Tự đông thay đổi hình ảnh,..

JavaScript, HTML, CSS là một trong ba thành phần cần phải học để làm được một website.

Không chỉ sử dụng để làm web, nó còn sử dụng một số ứng dụng khác. Ví dụ làm game bằng Unity…

# Nhúng JS vào HTML

JS được chèn trong trang web có một số cách. Tuy nhiên trong tài liệu sẽ đề câp hai cách dùng phổ biến:

* Chèn mã JS trong cặp thẻ <script></script>.
* Chèn file “.js” bởi cặp thẻ <script></script> qua thuộc tính “src”.

Chú ý rằng cặp thẻ <script></script> có thể đặt ở <head></head> hoặc <body></body>, tùy vào cách lâp trình.

## Nhúng trực tiếp mã JS

Để hiểu rõ chúng ta xem ngay ví dụ:

|  |
| --- |
| <!DOCTYPE html> <html> <body>   <button type="button" onclick="myFunction()">Change HTML</button> <p id=”demo”>HTML</p> <script> function myFunction() {    document.getElementById("demo").innerHTML = "New HTML."; } </script>  </body> </html> |

Button trong ví dụ khi click gọi tới thuộc tính onclick="myFunction()". Khi ấy sẽ gọi tới hàm myFunction() trong thẻ cặp thẻ <script></script>. Cặp thẻ script được nhúng trong thẻ body của web page.

Ta có thể chèn cặp thẻ trong cặp thẻ <head></head> hoặc cặp thẻ <body></body>.

## Nhúng JS bằng file .js

Cũng tương tự như CSS ta có thể chèn JS bằng file với đuô file .js.

Để hiểu rõ chúng ta tìm hiểu ví dụ sau:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <button type="button" onclick="myFunction()">Change HTML </button>  <p id=”demo”>HTML</p>  <p>(myFunction được gọi trong file "myScript.js")</p>  <script src="myScript.js"></script>  </body>  </html> |

Hàm “myFunction” nằm trong “myScript.js”. Như vậy chúng ta có thể viết tất cả mã JS vào trong file này, khi nhúng ta chỉ gọi tới file ta cần.

# Một số thành phần cơ bản của JS

## Biến

Biến trong các ngôn ngữ lập trình hay JS là một giá trị dữ liệu có thể thay đổi, khi ta tác động bằng mã nguồn.

Trong các ngôn ngữ lập trình JS thì phân hai loại biến: biến toàn cục và biến cục bộ.

Biến toàn cục là biến có thể sử dụng trong toàn ứng dụng.

Biến cục bộ là biến chỉ có thể sử dụng trong một khối của ứng dụng.

Mội số ví dụ cách khai báo biến JS như sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Ví dụ |  | | var x = 5; | Gán biến x với giá trị là 5 | | var y = 6; | Gán biến y bằng 6 | | x = x + 5; | Gán lại giá trị của x | | var z = x + y; | Gán giá trị của biến z bằng tổng của 5 và 6 | | var person = "John Doe"; | Gán biến bằng chuỗi | | var item; | Khởi tạo biến và giá trị là *undefined* | | element = 6; | Khai báo biến toàn cục element | | var length = 16; // Number | Khai báo biến kiểu số | | var lastName = "Johnson"; // String | Khai báo biến kiểu String | | var x = {firstName:"John", lastName:"Doe"}; // Object | Khai báo biến kiểu object | |

Khi đặt tên biến chú ý các quy tắc sau:

* Tên biến trong javascript phải là tên duy nhất.
* Tên có thể chứa chữ cái, chữ số, dấu gạch dưới và ký hiệu đô la.
* Tên phải bắt đầu bằng một chữ cái hoặc $ hoặc “\_”.
* Tên biến không được trùng từ khóa của JavaScript.

Lưu ý hơn nữa. Ngoài sử dụng biến *var* còn có cách khác khai báo biến như sau:

* Sử dụng const: const A = 5; Khai báo như này A sẽ là một hằng số không đổi. Nếu ta viết thêm A = 10; thì sẽ báo lỗi “Assignment to constant variable.”
* Sử dụng let. Let sẽ giới hạn phạm vi dùng biến. dùng nó khi dòng lệnh nằm trong cùng một khối.

## Kiểu dữ liệu

Các kiểu dữ liệu trong JS được phân loại ở bảng sau:

|  |  |
| --- | --- |
| Tên | Mô tả |
| boolean |  |
| null |  |
| undefine |  |
| number | có giá trị từ -(253 -1) đến 253 -1.  Để kiểm tra biên của giá trị JS cung cấp:  [Number.MAX\_VALUE](https://developer.mozilla.org/vi/docs/Web/JavaScript/Reference/Global_Objects/Number/MAX_VALUE) và [Number.MIN\_VALUE](https://developer.mozilla.org/vi/docs/Web/JavaScript/Reference/Global_Objects/Number/MIN_VALUE) |
| string | Hai hàm phổ biến: substring và concat() |
| symbol | Kiểu mới |
| object | Mảng cũng là kiểu object |

## Toán tử và số học JS

Sau đây là một số toán tử thường dùng khi viết JS:

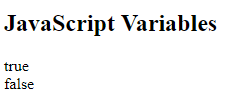
|  |  |
| --- | --- |
| Toán tử | Mô tả |
| X = y | So sánh bằng không quan tâm tới kiểu dữ liệu |
| X === y | So sánh bằng quan tâm tới kiểu dữ liệu |
| Typeof y | lấy kiểu dữ liệu của biếny |
| New date() | Toán tử khởi tạo |
| Person.name | Toán tử member |
| Person[“ name”] | Toán tử member |
| !(x==y) | Toán tử kiểm tra tính đúng đắn |
| + - \* / | Toán học cộng trừ nhân chia |
| % | Toán tử lấy số dư |
| ++ | Tăng dần |
| -- | Giảm dần |

Ngoài ra còn có các toán tử khác như phép dịch trái dịch phải,… tuy vậy rất ít khi dùng trong lập trình web.

Với toán tử so sánh == và === sẽ ra kết quả khác nhau nếu khác loại nhưng cùng giá trị.

Sau đây là ví dụ

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h2>JavaScript Variables</h2>  <p id="demo"></p>  <script>  var x= 0;  var y = "0";  document.write(x==y);  document.write('<br>');  document.write(x===y)  </script>  </body>  </html> |



H1: Kết quả sau khi chạy web.

Lưu ý:

* Khi cộng chuỗi với số JS coi như đang cộng 2 chuỗi với nhau : 16 + “String” tương đương với “16”+”String”.
* Khai báo biến kiểu ‘item = 5’ khi đó item sẽ là biến toàn cục sử dụng cho toàn ứng dụng
* *Undefined* là trường hợp đặc biệt, nên chú ý khi viết mã JS. Có thể rất dễ báo lỗi.

## Function trong JS

Cách khai báo hàm trong JS có các cách sau:

|  |
| --- |
| Cách khai báo |
| functionName = new function(parameter1, parameter2){  //logic  } |
| function functionName(parameter1, parameter2){  //logic  } |

Cách gọi function rất đơn giản: functionName(value1, value2,..);

## Object trong JS

Khai báo object có rất nhiều cách viết tuy vậy cùng chung một logic. Có một số cách viết như sau:

|  |  |
| --- | --- |
| Ví dụ cú pháp | Mô tả |
| var person = {}; // hoặc var person = new Object();  person.name = ‘’;  person.address = ‘’;  person.age = 0;  person.phone = ‘’; |  |
| var person = {  name: ‘’,  address: ‘’,  phone: ‘’,  email: ‘’,  age: ‘’;  getName: function(){  return this.name;  },  setName: function(){  // logic  }  }; | Nếu muốn thêm một trường mới thì vẫn phải dùng cách một, tuy nhiên cách code này sẽ quản lý code và rất dễ đọc và sửa chữa.  Cách sử dụng hàm trong đối tượng ta dùng this như code bên cạnh.  Khi ta thêm thuộc tính trong object |

## Mảng JS

Mảng là một biến đặc biệt, có thể chứa hơn một giá trị vào một thời điểm. Mảng là một tập hợp các nhóm cùng một kiểu dữ liệu.

Bản chất mảng là một object. Và typeof của mảng là object.

Đối với mảng có thể khai báo như sau:

|  |  |
| --- | --- |
| Cách viết | Mô tả hoặc lưu ý |
| var name\_array = [];  var name\_array = [1,2,3]; | Khai báo các phần tử khi khởi tạo |
| var name\_array = new Array();  var name\_array = new Array(1,2,3); | Sử dụng toán tử new |
| var person = ["John", "Doe", 46]; | Tương đương  var person = {firstName:"John", lastName:"Doe", age:46}; |

Cách add thêm một phần từ sử dụng hàm : push(…);

Ví dụ:

|  |
| --- |
| var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.push("Lemon");                // adds a new element (Lemon) to fruits |

Một số thuộc tính hoặc hàm sử dụng với mảng JS:

* push (..) để add thêm thêm phần tử vào mảng.
* sort() để sắp xếp mảng
* length để đếm tất cả các phần tử của mảng
* tên mảng[index] để lấy ra phần tử thứ index của mảng.

Ngoài cách sử dụng push(…) để add thêm phần tử trong mảng ta có thể sử dụng

|  |
| --- |
| fruits[fruits.length] = "Banana2"; |

Khi đó phần tử cuối cùng trong mảng sẽ là phần tử được thêm vào.

Lưu ý:

* Khi sử dụng toán tử new để tạo mảng là không tốt ví dụ

|  |  |
| --- | --- |
| var item = new Array(40, 100) | Tạo ra hai phần tử 40 và 100 |
| var item = new Array(40, 100); //giả sử phần tử 40 bị remove  var item = new Array( 100); | Khi này mảng sẽ là 100 phần tử |

# Nguồn

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects>

<https://www.w3schools.com/js/default.asp>