KHOA CÔNG NGHỆ THÔNG TIN – ĐH SƯ PHẠM HÀ NỘI
Bộ môn Kỹ thuật máy tính

NỀN TẢNG CN WEB

JAVASCRIPT

Giảng viên: Đỗ Ba Chín

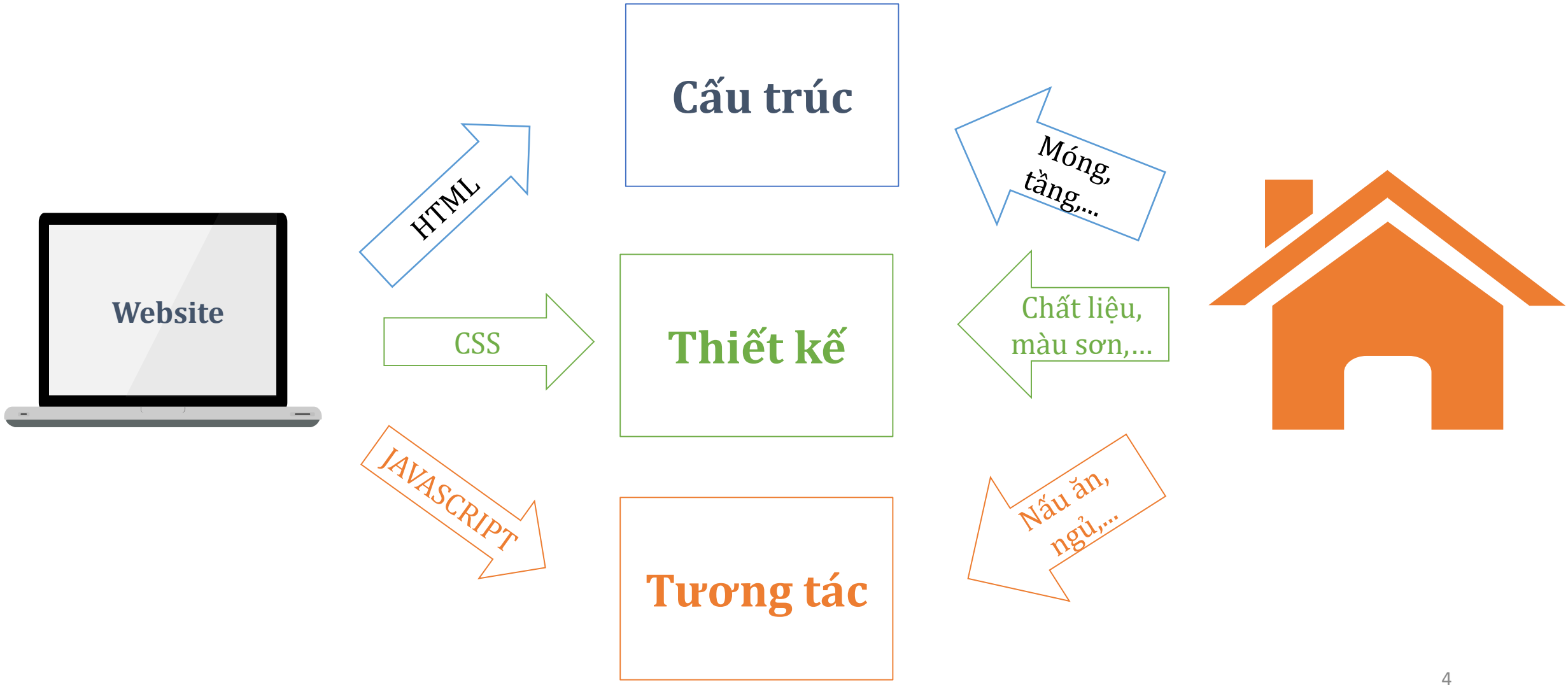
Hà Nội, 2022

TỔNG QUAN VỀ JAVASCRIPT

NỘI DUNG

- Tổng quan về Javascript
- Khai báo

TỔNG QUAN VỀ JAVASCRIPT



TỔNG QUAN VỀ JAVASCRIPT

- Là ngôn ngữ thông dịch (được dịch lúc chạy)
- Mã nguồn JS được nhúng hoặc tích hợp trực tiếp vào tập tin HTML
- Là ngôn ngữ lập trình frontend, giúp tăng tính tương tác của client với website
- Có thể tương tác với HTML, tương tác với client thông qua các sự kiện như click chuột, gõ phím,… làm cho trang HTML trở lên sinh động hơn

KHAI BÁO

- **Trong thẻ HTML thông qua các sự kiện (Inline)**

  VD: Click vào đây để show popup

- **Trong file HTML (Internal)**

  VD: **<script type="text/javascript">**

  // code javascript

  </script>

- **Link file .js bên ngoài (External)**

  VD: **<script type="text/javascript" src="js/main.js"></script>**

KHOA CÔNG NGHỆ THÔNG TIN – ĐH SƯ PHẠM HÀ NỘI
Bộ môn Kỹ thuật máy tính

NỀN TẢNG CN WEB

JAVASCRIPT CƠ BẢN

Giảng viên: Đỗ Ba Chín

Hà Nội, 2022

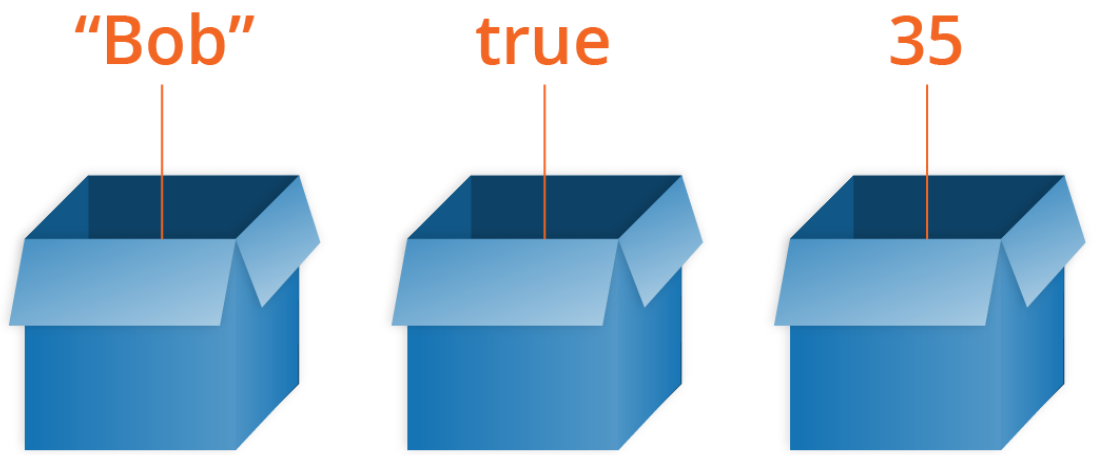
BIẾN

NỘI DUNG

- Tổng quan về biến
- Cú pháp khai báo biến
- Kiểu dữ liệu của biến

BIẾN

- Về mặt vật lý là 1 tham chiếu đến bộ nhớ trong thiết bị
- Biến chứa các giá trị có thể thay đổi được



CÚ PHÁP KHAI BÁO BIẾN

- Sử dụng từ khóa **var, let, const**

  VD: **var** username = ‘chin’ ;

- Quy tắc đặt tên
  - Tên biến phải bắt đầu 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 của JS

KIỂU DỮ LIỆU CỦA BIẾN

- 6 kiểu dữ liệu
  - Number //123, 1.23
  - String // ‘abc’, “123”
  - Boolean //true, false
  - Undefined //undefined
  - Array //[1, 2, 3, 'a']
  - Object //{name: ‘Chin', age: 23}
- Có thể sử dụng hàm **typeof** <variable> để kiểm tra kiểu dữ liệu của biến
- Các kiểu dữ liệu nguyên thủy là number, string, boolean
- **Chú ý**: kiểu dữ liệu array trong Javascript về bản chất vẫn là các object

KIỂU DỮ LIỆU MẢNG

- **Khai báo mảng:**

Cách 1: var name_array = [1,2,3]; *Cách 2*: var name_array = new Array(1,2,3);

- **Truy xuất các phần tử trong mảng**

tenmang[vitri] //trong đó vị trí của mảng bắt đầu từ 0

- **Duyệt mảng**

- Hàm length trả về số phần tử của mảng (name_array.length)
- VD Duyệt mảng

```
var name_array = [1,2,3];
for (var i = 0; i < name_array.length; i++){
   document.write(name_array[i]);
}
```

KIỂU DỮ LIỆU OBJECT

- **Khai báo object**

Cách 1: Sử dụng từ khóa new Object() : Var <tên object> = new Object();

Cách 2: Sử dụng từ khóa {} và thêm phương thức ngay lúc khai báo

Var <tên object> = { <thuộc tính> : <giá trị>;

<tên phương thức> : function() { //}

}

- **Lấy giá trị cho thuộc tính**

<tên object>.<tên thuộc tính>

- **Gán giá trị cho thuộc tính**

<tên object>.<tên thuộc tính> = giá trị;

HÀM

NỘI DUNG

- Hàm
- Phân loại hàm
- Gọi hàm
- Một số hàm cơ bản

HÀM

- Là 1 tập các dòng lệnh dùng để xử lý tác vụ nào đó
- Là khái niệm rất quan trọng trong lập trình
- Sử dụng hàm cho phép chia nhỏ chương trình thành các chức năng nhỏ hơn
- Tính chất quan trọng nhất của hàm là tính tái sử dụng

PHÂN LOẠI HÀM

- **Hàm xây dựng sẵn**

VD: alert(), prompt()

- **Hàm tự định nghĩa**

```
function function_name(parameter_list){
  //code Javacsript
  return value;
}
```

+ function_name: tên hàm do bạn tự định nghĩa
+ parameter_list: danh sách các tham số truyền vào hàm nếu có
+ return value: câu lệnh return này là tùy chọn, để thể hiện hàm này có mang giá trị nào không

GỌI HÀM

- Khi định nghĩa ra 1 hàm, thì hàm sẽ chưa được sử dụng cho đến khi nó được gọi
- Cú pháp

function_name(parameter_list);

- Ví dụ

```
function sum(a, b) {
  return a + b;
}
document.write("Tổng của 2 vs 5 = " + sum(2, 5));
```

MỘT SỐ HÀM CƠ BẢN

- *document*.write('<Nôi-dung>'); //hiển thị text Noi-dung lên màn hình
- console.log('<Nôi-dung>'); //hiển thị text Noi-dung tại tab Console khi Inspect trình duyệt, dùng để debug
- *alert*('<Nôi-dung>'); //hiển thị text Noi-dung dưới dạng popup của trình duyệt
- *prompt*('<Nôi-dung>'); //hiển thị text Noi-dung dưới dạng popup nhập liệu
- *confirm(*'<Nôi-dung>'); //hiển thị text Noi-dung dưới dạng popup xác nhận có thực thi hành động hay không
- *window.localtion = "<url>";* //chuyển hướng tới địa chỉ là url

PHẠM VI CỦA BIẾN

NỘI DUNG

- Phạm vi của biến
- Phạm vi cục bộ
- Phạm vi toàn cục
- Phân biệt var, let, const khi khai báo biến

PHẠM VI CỦA BIẾN

- Là vòng đời của 1 biến từ khi nó sinh ra cho đến khi chết đi
- Có 2 phạm vi chính:

- Cục bộ / Function (local) scope

- Toàn cục/ Global scope

PHẠM VI CỤC BỘ

Khi một biến được định nghĩa bên trong một hàm, biến đó được sử dụng local cho function đó và không thể được sử dụng bên ngoài

VD:

```
function local() {
  let number = 1;
  console.log(number); // 1
}
console.log(number); // undefined
```

PHẠM VI TOÀN CỤC

Khi một biến được khai báo bên ngoài hàm, thì có thể sử dụng bất cứ khi nào và bất cứ ở đâu

```
var number = 1;
function local() {
  console.log(number); // 1
}
console.log(number); // 1
```

PHÂN BIỆT TỪ KHÓA VAR, LEFT, CONST

	var	left	const
Phạm vi sử dụng	Toàn cục / Cục bộ	Cục bộ	Cục bộ

Tái khai báo	Có
	var x = 10;
	//...
	var x = 15; //Khai báo lại, không có lỗi gì
		Không
		let a = 20;
		//...
		let a = 100; //phát sinh lỗi, a đã khai báo
			Không

Cập nhật giá trị mới thay thế	Có	Có	Không

Trong trường hợp code lên đến hàng trăm, hàng nghìn dòng code, không biết được giá trị của biến liệu có bị thay đổi ở đoạn code nào sẽ dẫn đến việc debug là vô cùng khó khăn

TOÁN TỬ

NỘI DUNG

- Toán tử số học
- Toán tử so sánh
- Toán tử logic
- Toán tử gán

TOÁN TỬ SỐ HỌC

Các toán tử số học thực hiện tính toán trên dữ liệu dạng số (cụ thể hoặc biến).

Toán tử	Mô tả	Ví dụ
+	Phép cộng	A = 5 + 8
-	Phép trừ	A = 8 - 5
*	Phép nhân	A = 8 * 5
/	Phép chia	A = 20 / 5
%	Phép chia lấy số dư	10 % 3 = 1
++	Tăng lên một đơn vị	++x sẽ trả về giá trị của x sau khi tăng.x++ sẽ trả về giá trị của x trước khi tăng.
- -	Giảm một đơn vị.	--x sẽ trả về giá trị của x sau khi giảm. x-- sẽ trả về giá trị của x trước khi giảm. .

TOÁN TỬ SO SÁNH

Toán tử so sánh sử dụng trong các biểu thức về logic để so sánh bằng nhau, khác nhau. Giá trị trả về giá trị true hoặc false

Toán tử	Mô tả	Ví dụ
==	Bằng. Trả về giá trị true nếu các toán hạng bằng nhau.	a == b
!=	Không bằng. Trả về giá trị true nếu các toán hạng không bằng nhau.	a != 5
>	Lớn hơn. Trả về giá trị true nếu toán hạng trái lớn hơn toán hạng phải.	a > b
>=	Lớn hơn hoặc bằng. Trả về giá trị true nếu toán hạng trái lớn hơn hoặc bằng toán hạng phải.	a >= b >= c
<	Nhỏ hơn. Trả về giá trị true nếu toán hạng trái nhỏ hơn toán hoạng phải.	a < b
<=	Nhỏ hơn hoặc bằng. Trả vè giá trị true nếu toán hạng trái nhỏ hơn hoặc bằng toán hạng phải.	a <= b<= c

TOÁN TỬ LOGIC

Toán tử logic gồm các phép toán : Phép và, phép hoặc phép phủ định

Toán tử	Giá trị	Mô tả
And (&&)	expr1 && expr2	Trả về true khi cả 2 biểu thức expre1 và expre2 trả về true
Or (||)	expr1 || expr2	Trả về true khi có ít nhất 1 trong 2 biểu thức expre1 và expre2 trả về true
Not (!)	!expr	Trả về giá trị false nếu biểu thức đúng và trả về giá trị true nếu biểu thức sai

TOÁN TỬ GÁN

Toán tử gán được dùng để gán giá trị ở bên phải toán tử vào biến ở bên trái toán tử.

Toán tử	Ví dụ	Viết đầy đủ
=	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;

ĐIỀU KHIỂN RẼ NHÁNH

NỘI DUNG

- If
- If…else
- If…elseif…else
- Biểu thức switch…case

If

- Lệnh **if** : nếu điều kiện là đúng (true) thì thi hành các lệnh trong khối

  - nếu điều kiện sai (false) thì khối lệnh sau nó bị bỏ qua
- Chỉ kiểm tra duy nhất 1 trường hợp khi biểu thức điều kiện trả về giá trị TRUE
- **Cú pháp**

```
if (điều_kiện) {
 //Các dòng lệnh khi điều kiện trả về giá trị true
}
```

If ... else

- Nếu biểu thức logic là true thi hành các lệnh trong khối if, nếu false thì thi hành khối lệnh else
- **Cú pháp**

```
if (điều_kiện) {
 //Các dòng lệnh khi điều kiện trả về giá trị true
}
else{
      //Các dòng lệnh khi điều kiện trả về giá trị false
}
```

if... elseif... else

- else if sẽ tạo ra câu lệnh điều kiện if mới nếu điều kiện trước đó false
- **Cú pháp**

if (điều_kiện 1) { //Các dòng lệnh khi điều kiện 1 trả về giá trị true}

elseif (điều_kiện 2) {//Các dòng lệnh khi điều kiện 2 trả về giá trị true}

elseif (điều_kiện 3) {//Các dòng lệnh khi điều kiện 3 trả về giá trị true}

……

else {//Các dòng lệnh khi điều kiện trả về giá trị false}

BIỂU THỨC SWITCH…CASE

- Trong trường hợp có rẽ nhánh (nhiều điều kiện) khác nhau có thể thay thế if..elseif..else bằng switch … case
- **Cú pháp**

```
switch (expression) {
  case n1:
     //.. thi hành nếu expression bằng n1
    break;
  case n2:
     //.. thi hành nếu expression bằng n2
    break;
……
  default:
    //.. mặc định thi hành nếu expression không bằng giá trị nào ở trên
}
```

Sử dụng để điều hướng ra khỏi khối, nếu không có break thì sẽ không thoát lệnh mà sẽ thi hành khối tiếp theo

Định nghĩa khối mặc định, khối này thi hành nếu tất cả các điều kiện rẽ nhánh không thỏa mãn

VÒNG LẶP

NỘI DUNG

- For
- While
- Do…While
- Từ khóa break - continue

For

- **Cú pháp**

```
for (initialization; test condition; iteration statement) {
  //Khối lệnh thi hành
}
```

lệnh thi hành trước khi vòng lặp for bắt đầu (khởi tạo)

Đều kiện kiểm tra trước mỗi lần thi hành khối lệnh for (true thì khối lệnh sẽ thi hành, false sẽ khối for sẽ không thi hành - thoát lặp)

Thi hành sau mỗi lần một vòng hoàn thành

- Thường sử dụng khi biết trước số lần lặp
- Cần chú ý về điều kiện dừng vòng lặp, tránh vòng lặp vô hạn

While

- **Cú pháp**

  while (điều_kiện) {

  //Khối lệnh thi hành

  }

- **Mô tả:** Kiểm tra điều kiện, nếu true sẽ thi hành khối lệnh. Đến cuối khối lại kiểm tra điều kiện, nếu điều kiện vẫn là true thì lại tiếp tục thì hành vòng mới của khối lệnh.
- Thường sử dụng khi không biết trước số lần lặp
- Cần chú ý về điều kiện dừng vòng lặp, tránh vòng lặp vô hạn

Do...While

- **Cú pháp**

  do {

  //Khối lệnh thi hành

  }

  while (điều_kiện)

- Bản chất giống vòng lặp while nhưng khối lệnh thi hành luôn mà không kiểm tra điều kiện trước, khi khối lệnh thi hành xong mới kiểm tra điều kiện để xem có lặp lại hay không
- Khác biệt duy nhất là vòng lặp luôn thực hiện logic code ít nhất 1 lần, cho dù điều kiện while là FALSE
- Cần chú ý về điều kiện dừng vòng lặp, tránh vòng lặp vô hạn

Từ khóa Break - continue

- Break: bỏ qua các lệnh còn lại đồng thời thoát khỏi vòng lặp
- Continue: bỏ qua các lệnh còn lại và khởi tạo vòng lặp mới luôn

DOM

NỘI DUNG

- Tổng quan về DOM
- Nhiệm vụ của DOM Javascript
- Các loại DOM trong Javascript

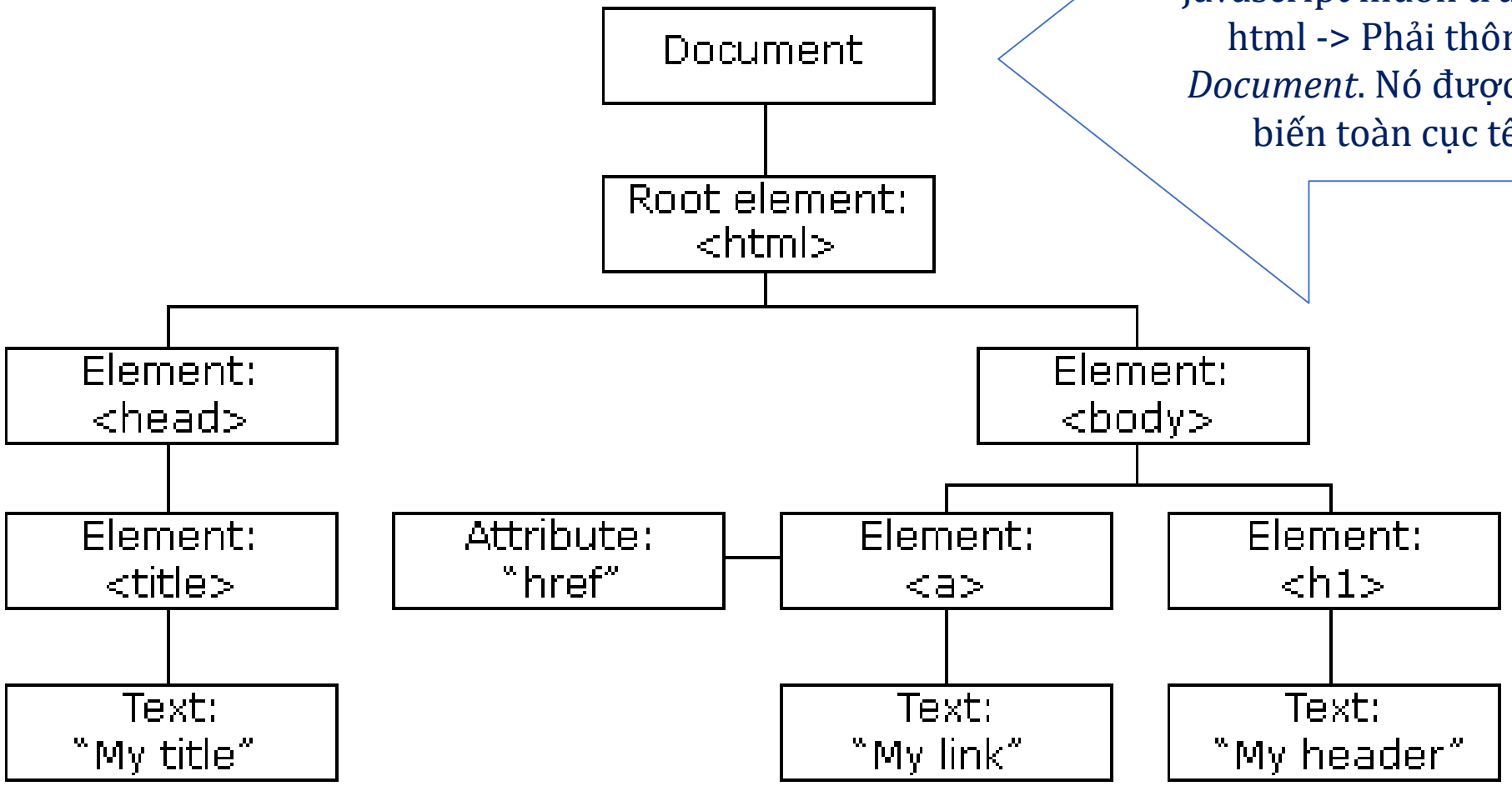
Tổng quan về DOM

- **DOM** (**D**ocument **O**bject **M**odel): Mô hình các đối tượng trong tài liệu HTML.
- Khi trang web được tải → Trình duyệt (browser) dựa trên các quy chuẩn (W3C – Word Wide Web Consortium) đã được cài đặt sẵn → dịch tài liệu (đoạn mã HTML) → Tạo ra một mô hình DOM

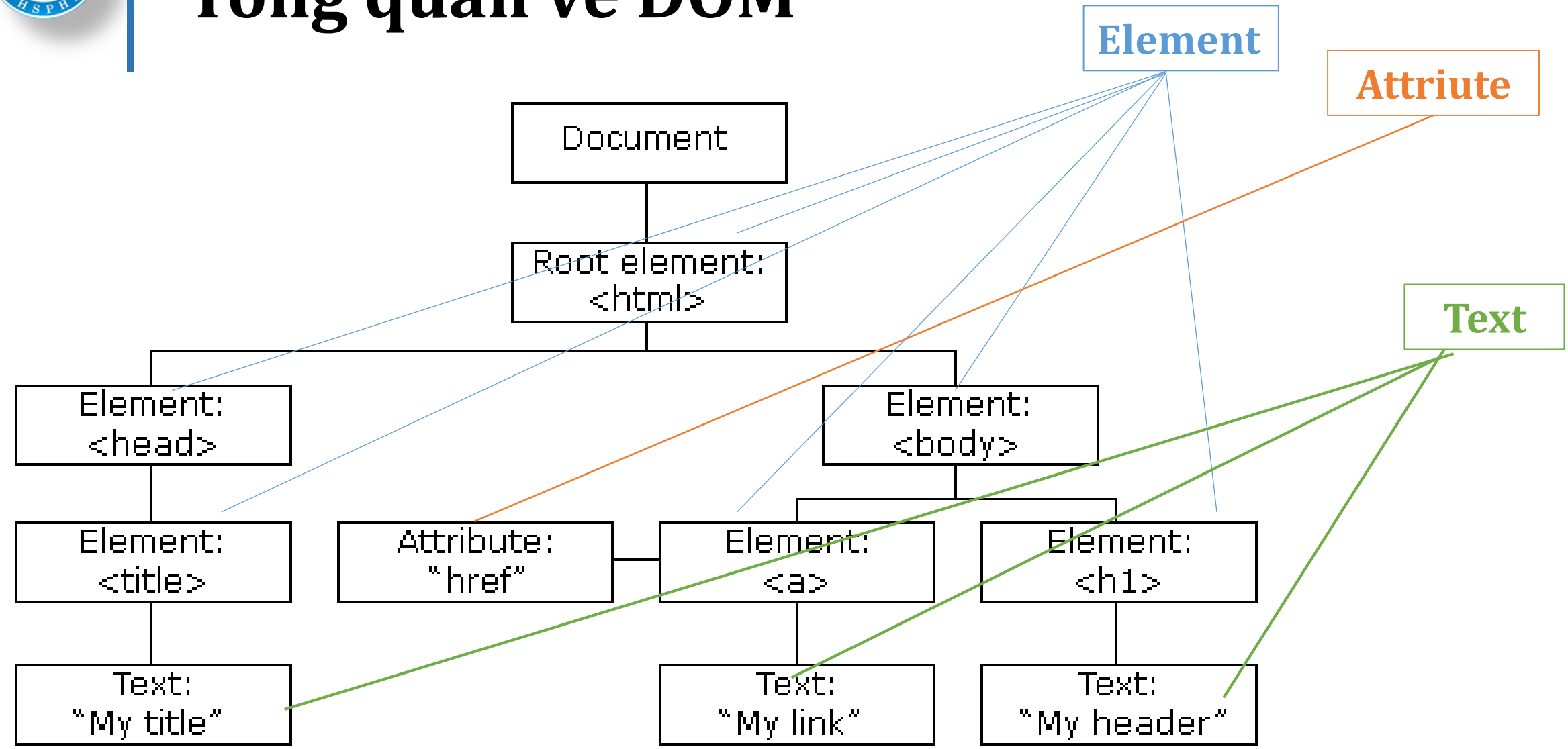
→ **DOM** sẽ có nhiệm vụ xử lý các vấn đề như đổi thuộc tính, đổi cấu trúc,.... của các thẻ HTML

Tổng quan về DOM

Javascript muốn truy xuất đến một thẻ html -> Phải thông qua *đối tượng Document*. Nó được lưu trữ trong một biến toàn cục tên là *document*.



Tổng quan về DOM



Tổng quan về DOM

Mô hình **DOM** sẽ gồm 3 thành phần chính:

- **Element (thành phần):** Tất cả các thẻ (tag) trong file tài liệu được sử dụng
- **Attribute:** Các thuộc tính nằm trong thẻ mở
- **Text:** các đoạn chữ không được đặt trong thẻ nào haowcj đặt giữa 2 thẻ đóng mở

Nhiệm vụ của DOM Javascript

- Cho phép truy xuất đến các thẻ html.
- Thay đổi các thuộc tính của thẻ html
- Thay đổi css của các thẻ html
- Tạo, xóa, thêm các thẻ html

Các loại DOM trong Javascript

Xét về tính chất DOM chia ra làm 8 loại khác:

- **DOM document**: có nhiệm vụ lưu trữ toàn bộ các thành phần trong tài liệu của website
- **DOM element**: có nhiệm vụ truy xuất tới thẻ HTML nào đó thông qua các thuộc tính như tên class, id, name của thẻ HTML
- **DOM HTML**: có nhiệm vụ thay đổi giá trị nội dung và giá trị thuộc tính của các thẻ HTML
- **DOM CSS**: có nhiệm vụ thay đổi các định dạng CSS của thẻ HTML
- **DOM Event**: có nhiệm vụ gán các sự kiện như onclick(), onload() vào các thẻ HTML
- **DOM Listener**: có nhiệm vụ lắng nghe các sự kiện tác động lên thẻ HTML đó
- **DOM Navigation** dùng để quản lý, thao tác với các thẻ HTML, thể hiện mối quan hệ cha - con của các thẻ HTML
- **DOM Node, Nodelist**: có nhiệm vụ thao tác với HTML thông qua đối tượng (Object)

CHỌN PHẦN TỬ TRONG DOM

NỘI DUNG

- Lấy Element node
- Thêm Element node
- Attribute node
- Text node

Lấy Element

- ✓ Tìm 1 phần tử theo id (trả về HTMLElement hoặc null nếu không thấy)

  Cú pháp: document.getElementById('idname')

- ✓ Tìm các phần tử theo class (trả về HTMLCollection)

  Cú pháp: document.getElementsByClassName('classname')

- ✓ Tìm các phần tử theo tên thẻ (trả về HTMLCollection)

  Cú pháp: document.getElementsByTagName('tagname')

- ✓ Tìm các phần tử theo CSS Selector (trả về HTMLCollection)

  Cú pháp: document.qerySelector('selector')

Thêm Element

B1: Lấy ra Element cần thêm

B2:

+) Lấy Element con

Cú pháp <Elemmet cần thêm>.innerHTML = ‘<Elemnet cần thêm>’ ;

+) Ghi đè Element con

Cú pháp <Elemmet cần thêm>.outerHTML = ‘<Elemnet cần thêm>’ ;

Attribute

- **Lấy ra Attribute:** Lấy ra Element -> Lấy Atrribute

Cách 1 Trực tiếp <Element đã lấy>.<Attribute cần lấy>

Cách 2 Tùy biến <Element đã lấy>.getAttribute('thuộc tính');

- **Thêm Attribute:** Lấy ra Element -> Gán Atrribute

Cách 1 Trực tiếp qua thuộc tính: <Element đã lấy>.<Attribute cần thêm> = 'Giá trị';

Cách 2 Thông qua phương thức (sử dụng trong trường hợp thuộc tính không phải của thẻ tương ứng, tùy biến) : <Element đã lấy>.setAttribute('thuộc tính', giá trị');

Text node

- **Lấy text node:** Lấy ra Element -> Lấy text node

Cách 1: <Element đã lấy>.innerText ;

Cách 2 : <Element đã lấy>.textContent ;

- **Gán text node:** Lấy ra Element -> Gán text node

Cách 1: <Element đã lấy>.innerText = 'giá trị'; // lấy text trực tiếp text của Elemnet

Cách 2 : <Element đã lấy>.textContent = 'giá trị'; //lấy text của cả các con

DOM EVENT

Tổng quan về Event trong Javascript

- Là một hành động tác động lên các đối tượng HTML, qua đó ta có thể bắt được sự kiện và yêu cầu javascript thực thi một chương trình nào đó
- Mỗi sự kiện có thể gán nhiều hành động, tùy thuộc vào từng chức năng cụ thể
- **Ví dụ:** Có Form đăng ký tài khoản và muốn bắt sự kiện khi người dùng CLICK vào button đăng ký thì hiện những hành động như:
  - Kiểm tra người dùng có nhập dữ liệu không.
  - Kiểm tra người dùng nhập dữ liệu có đúng định dạng không.
  - ...

Thêm / bắt sự kiện trong Javascript

- **Cách 1: Bắt trực tiếp trong thẻ HTML**

Ví dụ:

```
<a href="#" onclick="do_something()"> Khoa công nghê thông tin </a>


function do_something(){
    alert(‘Chào mừng bạn đã đến với khoa CNTT!');
}
```

Thêm / bắt sự kiện trong Javascript

- **Cách 2: Bắt sự kiện cho một thẻ HTML bằng javascript**

Cú pháp:

```
<elementObject>.<eventName>  = function(){
   // do something
};
```

Trong đó

- <elementObject> là đối tượng HTML sử dụng **DOM** để lấy
- <eventName> là tên của event như onclick, onchange, ...

Thêm / bắt sự kiện trong Javascript

- **Cách 2: Bắt sự kiện cho một thẻ HTML bằng javascript**

Ví dụ:

```
<input type="button" id="show-btn" value="Click me" />
<script type="text/javascript“>
    var button = document.getElementById("show-btn");
    button.onclick = function()
            {
             alert("Bạn vừa click vào button");
            };
</script>
```

Lấy dối tượng

Thêm sự kiện cho đối tượng

Các sự kiện (Events) trong javascrpt

STT	Event Name	Description (Mô tả)
1	onclick	Xảy ra khi click vào thẻ HTML
2	ondbclick	Xảy ra khi double click vào thẻ HTML
3	onchange	Xảy ra khi giá trị (value) của thẻ HTML đổi. Thường dùng trong các đối thẻ form input
4	onmouseover	Xảy ra khi con trỏ chuột bắt đầu đi vào thẻ HTML
5	onmouseout	Xảy ra khi con trỏ chuột bắt đầu rời khỏi thẻ HTML
6	onmouseenter	Tương tự như onmouseover
7	onmouseleave	Tương tự như onmouseout
8	onmousemove	Xảy ra khi con chuột di chuyển bên trong thẻ HTML

Các sự kiện (Events) trong javascrpt

STT	Event Name	Description (Mô tả)
9	onkeydown	Xảy ra khi gõ một phím bất kì vào ô input
10	onload	Sảy ra khi thẻ HTML bắt đầu chạy, nó giống như hàm khởi tạo trong lập trình hướng đối tượng
11	onkeyup	Xảy ra khi gõ phím, khi lúc nhả phím ra sẽ được kích hoạt
12	onkeypress	Xảy ra khi nhấn môt phím vào ô input
13	onblur	Xảy ra khi con trỏ chuột rời khỏi ô input
14	oncopy	Xảy ra khi copy nội dung của thẻ
15	oncut	Xảy ra khi cắt nội dung của thẻ
16	onpaste	Xảy ra khi dán nội dung vào thẻ

TỔNG KẾT

- Tổng quan về Javascript
- DOM

Thanks