

Unity
Thống nhất
Excellence
Vượt trội
Leadership
Tiên phong

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT



**PHÁT TRIỂN WEB
KINH DOANH**

GV: *ThS Nguyễn Quang Phúc*

Phát triển web kinh doanh: Javascript & DOM

ThS. Nguyễn Quang Phúc
phucnq@uel.edu.vn

NỘI DUNG

01

Giới thiệu về Javascript

02

Lập trình Javascript

03

Cấu trúc Document Object Model (DOM)

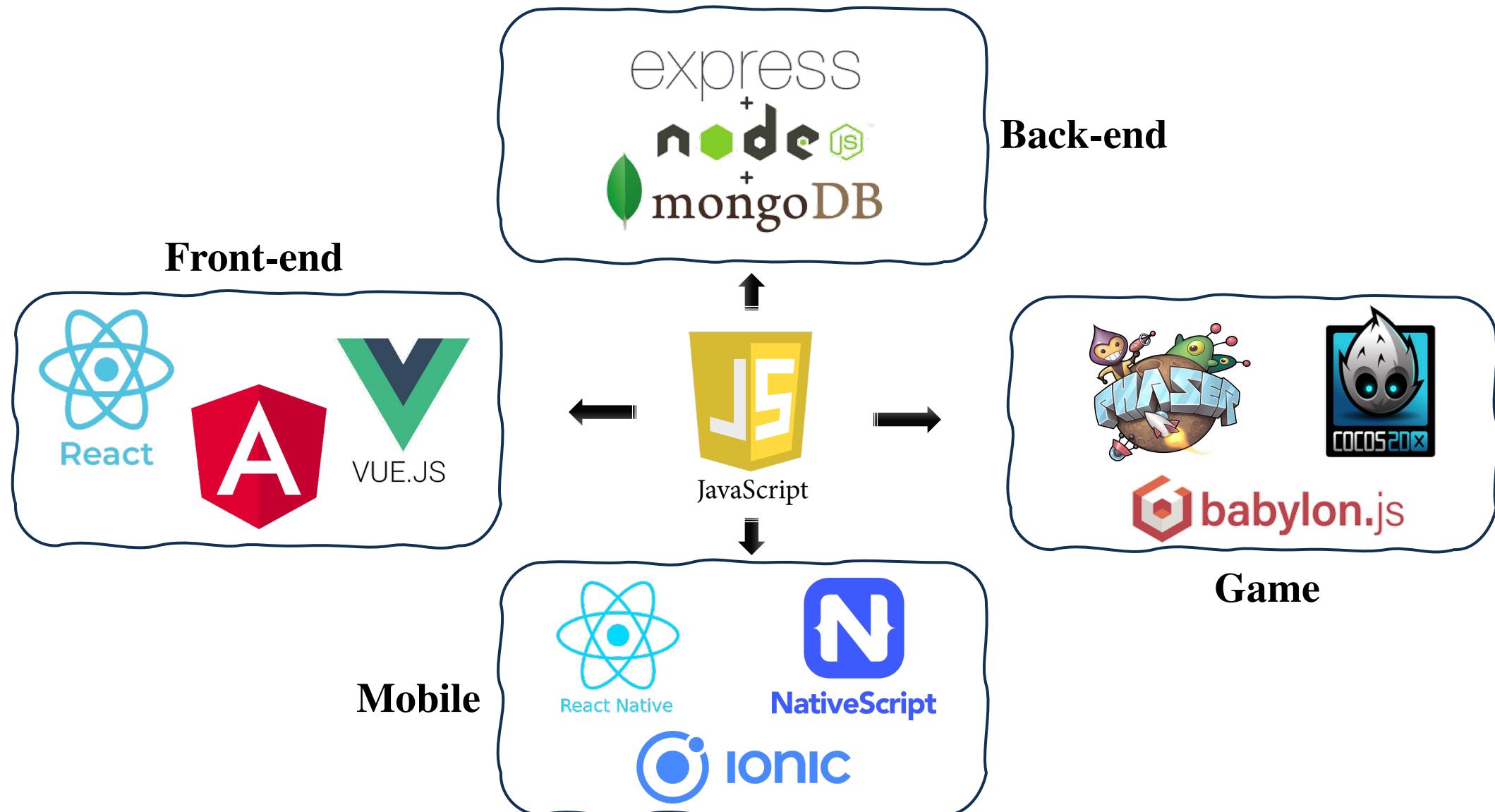
1. Giới thiệu về Javascript

- Javascript là ngôn ngữ lập trình kịch bản để tạo script từ phí client (client-side) cho phép: xử lý tương tác người dùng, thay đổi nội dung động, xác nhận tính hợp lệ dữ liệu,
- Javascript được tạo ra bởi Brendan Eich (một nhân viên của Netscape) vào tháng 09/1995, sau đó được đổi tên thành Mocha → Mona → Livescript → Javascript.
- Tính đến năm 2016, có đến 92% trang web đang sử dụng Javascript.
- Tiêu chuẩn: ECMAScript, phiên bản hiện tại ES12 (ES2021)

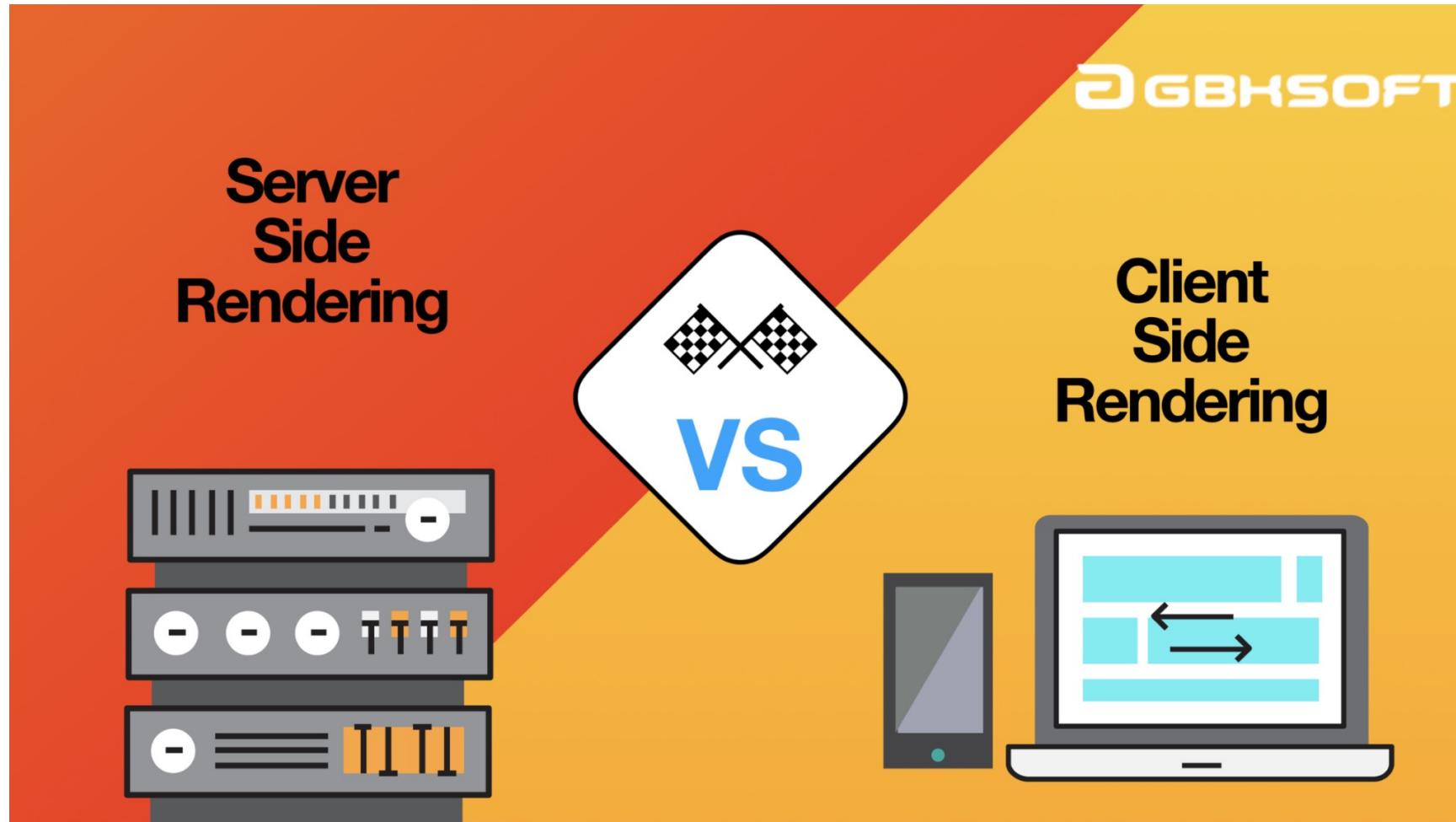


Brendan Eich
Dob: July 4, 1961

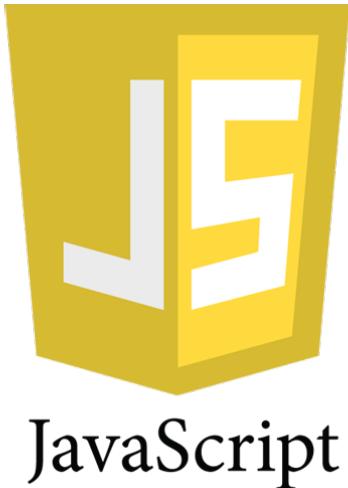
1. Giới thiệu về Javascript



2. Lập trình Javascript



2. Lập trình Javascript

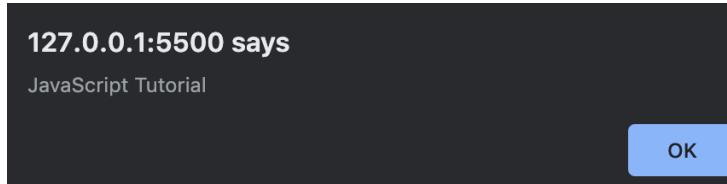


- Change HTML Content
- Change HTML Attribute Values
- Change HTML Styles (CSS)
- Hide/Show HTML Elements

2. Lập trình Javascript

»» <script> </script>

- ✓ Mã nguồn Javascript được đặt trong cặp thẻ <script> </script>



The image shows a screenshot of a browser window. On the left, there is a code editor window containing the following JavaScript code:

```
<script>
    alert("JavaScript Tutorial")
</script>
```

To the right of the code editor is a screenshot of a browser alert dialog box. The dialog box has a dark background and contains the text "127.0.0.1:5500 says" followed by "JavaScript Tutorial". In the bottom right corner of the dialog box, there is a blue "OK" button.

- ✓ Mã nguồn Javascript có thể được **tổ chức dạng hàm (function)** và được **gán vào sự kiện (event)** nhất định, hàm được thực thi khi sự kiện xảy ra.

```
<script>
    function sayHello() {
        document.getElementById('content').innerText = "Hello";
    }
</script>
```

```
<body>
    <button onclick="sayHello();">Say Hello</button>
    <div id="content"></div>
</body>
```

2. Lập trình Javascript

»» <script> </script>

- ✓ Thẻ **<script>** có thể được đặt trong thẻ **<head>** hoặc **<body>**
- ✓ Load external javascript

```
<script src="script.js"></script>
```

```
<head>
  <script>
    function sayHello() {
      document.getElementById('content').innerText = "Hello";
    }
  </script>
</head>
```

```
<body>
  <button onclick="sayHello();">Say Hello</button>
  <div id="content"></div>
</body>
```

≠ ???

```
<body>
  <button onclick="sayHello();">Say Hello</button>
  <div id="content"></div>

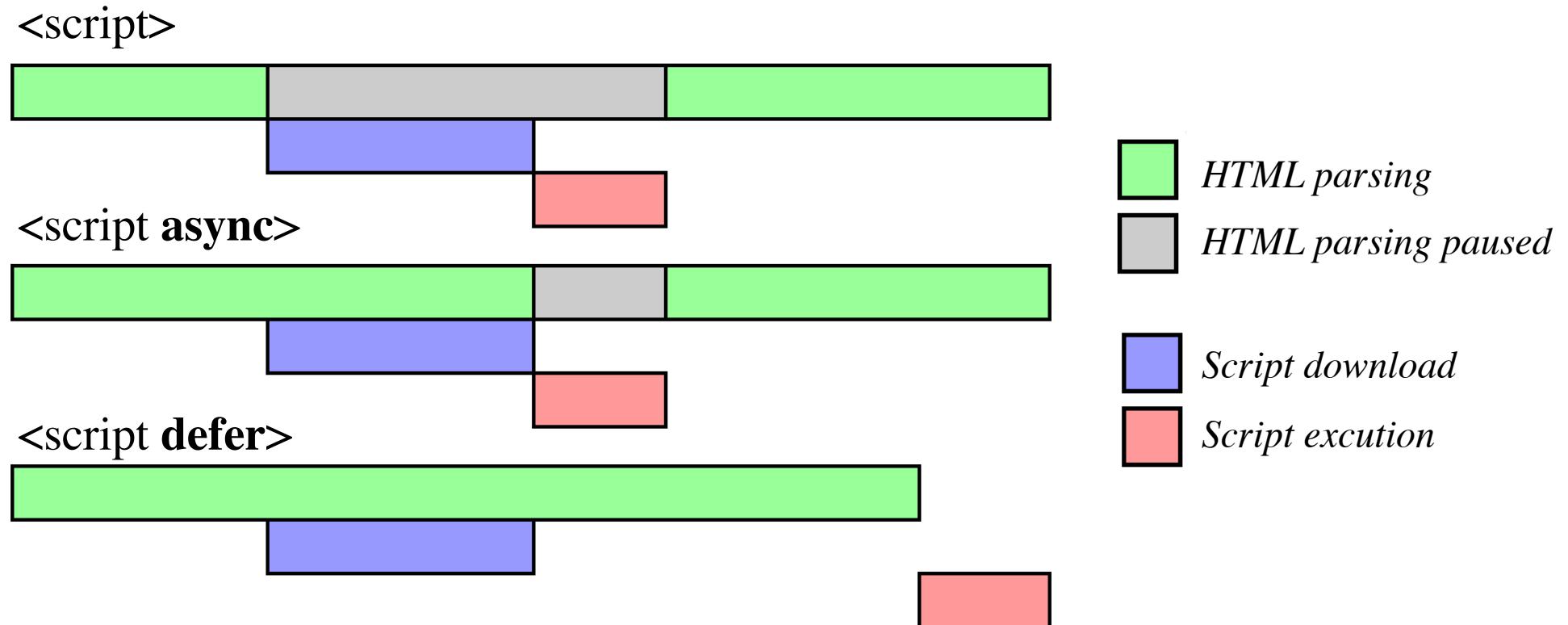
  <script>
    function sayHello() {
      document.getElementById('content').innerText = "Hello";
    }
  </script>
</body>
```

2. Lập trình Javascript

»» <script> </script>

- ✓ Cơ chế tải và thực thi Javascript

async vs defer



2. Lập trình Javascript

»» “Display” data

- ✓ Writing into an HTML element, using **innerHTML/innerText**

```
<h3>Welcome to: <span id="content"></span></h3>
<script>
|   document.getElementById("content").innerHTML = "Vietnam";
</script>
```

- ✓ Writing into the HTML output using **document.write()**

```
<span>Welcome to: </span>
<script>
|   document.write("Vietnam");
</script>
```

*Using `document.write()` after an HTML document is loaded, will **delete all existing HTML**

2. Lập trình Javascript

»» “Display” data

- ✓ Writing into an alert box, using **window.alert()**

```
<script>
|   window.alert("Welcome to Vietnam");
</script>
```

- ✓ Writing into the browser console, using **console.log()**

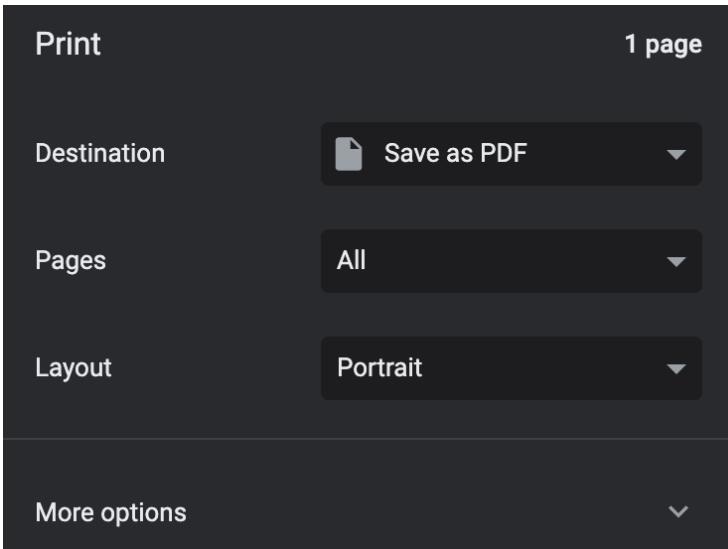
```
<script>
|   console.log("Welcome to Vietnam");
</script>
```

2. Lập trình Javascript

»» “Display” data

- ✓ Printing the content of the current window, using **window.print()** (method in the browser)

```
<h2>The window.print() Method</h2>
<p>Click the button to print the current page.</p>
<button onclick="window.print()">Print this page</button>
```



2. Lập trình Javascript

»» Statements

```
<script>
    let x, y, z; // Statement 1
    x = 10; // Statement 2
    y = 20; // Statement 3
    z = x + y; // Statement 4
    console.log("Sum: ", z); // Statement 5
    console.log("Sum: " + x + " + " + y + " = " +
                z); // Statement 6
</script>
```

»» Code blocks: { }, if ..else..., ...

```
{
    var x = 2,
        y = 9
    console.log(x);
}
```

2. Lập trình Javascript

»» Keywords

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

2. Lập trình Javascript

»» Reserved Words

In JavaScript you cannot use these reserved words as variables, labels, or function names:

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Words marked with* are new in ECMAScript 5 and 6.

2. Lập trình Javascript

»» Quy ước về cú pháp

- ✓ Câu lệnh JavaScript được kết thúc bằng dấu ; hoặc không
- ✓ Không phân biệt khoảng trắng, tab, xuống dòng trong câu lệnh
- ✓ JavaScript phân biệt chữ hoa, chữ thường

```
<script>
    let x, y;
    x = 10
    y = 20;
    z = x + y
    var Z = 50;
    console.log("Sum: " + x + " + " +
                y + " = " +
                z);
</script>
```

Sum: 10 + 20 = 30

2. Lập trình Javascript

»» Quy ước về cú pháp

- ✓ Dữ liệu chuỗi có thể được đặt trong nháy đơn hoặc nháy kép

```
console.log("Welcome to Vietnam");
alert('Welcome to "Vietnam"');
```

- ✓ Ghi chú với cú pháp:

```
// Ghi chú trên 1 dòng
/* Ghi chú trên
nhiều dòng */
```

2. Lập trình Javascript

»» Quy ước về cú pháp

✓ Tên biến và hàm:

- Bắt đầu bằng Ký tự (A..Z, a..z), _, \$
- Không bắt đầu bằng ký số (0..9)
- Không có khoảng trắng trong biến hoặc hàm
- Không đặt tên trùng từ khóa

✓ Quy chuẩn:

- **Underscore**: first_name, last_name, master_card, inter_city.
- **Upper Camel Case**: FirstName, LastName, MasterCard, InterCity.
- **Lower Camel Case**: firstName, lastName, masterCard, interCity.

2. Lập trình Javascript

» Biến

- ✓ **Biến** là “nơi lưu trữ” các giá trị xử lý trong quá trình thực thi chương trình.
- ✓ Trong Javascript có thể khai báo biến với các từ khóa sau: **var**, **let**, **const**.

```
var x = 8;  
var productName = "Heineken", carName = 'Mercedes';  
var checked = true;  
  
let x = 8, y = 10;  
let checked = false;  
  
const PI = 3.14;  
const cars = ["BMW", 'Volvo', "Mercedes"];
```

var ≠ let, const
var, let ≠ const



2. Lập trình Javascript

» Biến ➤ var

```
var productName; // --> undefined  
productName = "Corona Extra"; // --> Corona Extra
```

```
productName = "Corona Extra"; // --> Corona Extra  
productName = 'Sapporo';  
console.log(productName); // --> Sapporo
```

```
var productName = "Corona Extra"; // --> Corona Extra  
var productName;  
console.log(productName); // --> ???
```

```
if (true) {  
    var s = 'Welcome to Vietnam';  
    alert(s);  
}  
console.log(s); // --> Welcome to Vietnam
```

2. Lập trình Javascript

» Biến ➤ let

- Không thể khai báo lại biến được khai báo.
- Các biến phải được khai báo trước khi sử dụng.
- Các biến chỉ được sử dụng trong phạm vi khôi.

```
let productName = "Heineken";
let productName = "Tiger";
// SyntaxError: Identifier 'productName' has already
been declared
```

```
productName = "Heineken";
let productName;
console.log(productName);
// --> ReferenceError: Cannot access 'productName'
before initialization
```

```
if (true) {
  let s = 'Welcome to Vietnam';
  alert(s);
}
console.log(s); // --> Uncaught ReferenceError: s is
not defined at ...
```

2. Lập trình Javascript

» Biến ➤ const

- Không thể khai báo lại biến được khai báo. ($\Leftrightarrow let$)
- Không thể gán lại giá trị cho biến. ($\neq var, let$)
- Các biến chỉ được sử dụng trong phạm vi khôi. ($\Leftrightarrow let$)

```
const PI = 3.141592653589793;
const PI = 3.14; // Uncaught SyntaxError: Identifier
'PI' has already been declared
```

```
const PI = 3.141592653589793;
PI = 3.14; // This will give an error
PI = PI + 10; // This will also give an error
// --> Uncaught TypeError: Assignment to constant
variable.
```

```
{
  const PI = 3.14;
  alert(PI);
}

console.log(PI); // --> Uncaught ReferenceError: PI is
not defined ...
```

2. Lập trình Javascript

»» Biến ➤ const

```
const cars = ["Audi", "Volvo", "BMW"];
cars[0] = "Mercedes"; // You can change an element
cars.push("Lexus"); // You can add an element
console.log(cars);
cars = ["Inova", "Civic", "Sonata"]; // Not allowed
```

```
const car = {
    type: "Fiat",
    model: "500",
    color: "white"
};
car.color = "red"; // You can change a property
car.owner = "Johnson"; // You can add a property
console.log(car);
car = {type:"Volvo", model:"EX60", color:"red"}; // Not allowed
```

2. Lập trình Javascript

» Toán tử số học

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (<u>ES2016</u>)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

2. Lập trình Javascript

»» Toán tử gán

Operator	Example	Same As
=	$x = y$	$x = y$
$+=$	$x += y$	$x = x + y$
$-=$	$x -= y$	$x = x - y$
$*=$	$x *= y$	$x = x * y$
$/=$	$x /= y$	$x = x / y$
$%=$	$x \%= y$	$x = x \% y$
$**=$	$x **= y$	$x = x ** y$

2. Lập trình Javascript

»» Toán tử nối chuỗi (+)

```
let x = 10;
x += 5; // --> x = 15 (number)
let y = 8;
y += '8'; // --> y = 88
console.log(typeof(y)); // --> string
let car = "Lexus",
    model = 'LX570';
car += " " + model;
console.log(car); // Lexus LX570
```

2. Lập trình Javascript

»» Toán tử so sánh

Given that $x = 5$

Operator	Description	Comparing	Returns
==	equal to	$x == 8$	false
		$x == 5$	true
		$x == "5"$	true
===	equal value and equal type	$x === 5$	true
		$x === "5"$	false
!=	not equal	$x != 8$	true
!==	not equal value or not equal type	$x !== 5$	false
		$x !== "5"$	true
		$x !== 8$	true
>	greater than	$x > 8$	false
<	less than	$x < 8$	true
>=	greater than or equal to	$x >= 8$	false
<=	less than or equal to	$x <= 8$	true

»» Toán tử logic

Given that $x = 7$ and $y = 2$

Operator	Description	Example
&&	and	$(x < 10 \&\& y > 1)$ is true
	or	$(x == 5 y == 5)$ is false
!	not	$!(x == y)$ is true

»» Toán tử 3 ngôi (?)

variablename = (condition) ? value1 : value2

```
let voteable = (age < 18) ? "Too young" : "Old enough";
```

2. Lập trình Javascript

»» Conditional statements: if else and else if

```
if (condition1) {  
    // block of code to be executed if condition1  
    // is true  
} else if (condition2) {  
    // block of code to be executed if the  
    // condition1 is false and condition2 is true  
} else {  
    // block of code to be executed if the  
    // condition1 is false and condition2 is false  
}
```

```
const time = new Date().getHours();  
let greeting;  
if (time < 10) {  
    greeting = "Good morning";  
} else if (time < 20) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}  
console.log(greeting);
```

2. Lập trình Javascript

»»» Switch statement

```
switch (expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```

```
switch (new Date().getDay()) {  
    case 0:  
        day = "Sunday";  
        break;  
    case 1:  
        day = "Monday";  
        break;  
    case 2:  
        day = "Tuesday";  
        break;  
    case 3:  
        day = "Wednesday";  
        break;  
    case 4:  
        day = "Thursday";  
        break;  
    case 5:  
        day = "Friday";  
        break;  
    case 6:  
        day = "Saturday";  
}
```

2. Lập trình Javascript

»» Vòng lặp For

✓ Cú pháp:

```
for (Exp1; Exp2; Exp3)  
    statement
```

✓ Ý nghĩa:

- Exp1: biểu thức khởi tạo.
- Exp2: biểu thức điều kiện.
- Exp3: biểu thức điều khiển lặp.

```
let text = "";  
for (let i = 0; i < 5; i++) {  
    text += "The number is " + i + "<br>"  
}  
document.getElementById("demo").innerHTML = text;
```

```
const cars = ["BMW", "Volvo", "Mercedes"];  
let i, len, text;  
for (i = 0, len = cars.length, text = ""; i < len; i++) {  
    text += cars[i] + "<br>"  
}  
document.getElementById("demo").innerHTML = text;
```

```
const cars = ["BMW", "Volvo", "Mercedes"];  
let i = 0, len = cars.length, text = "";  
for (; i < len;) {  
    text += cars[i] + "<br>"  
    i++;  
}  
document.getElementById("demo").innerHTML = text;
```

2. Lập trình Javascript

»»» Vòng lặp For ... In

- ✓ Lặp qua các **thuộc tính** của đối tượng
- ✓ Cú pháp:

```
for (key in object) {  
    // code block to be executed  
}
```

```
const person = {  
    fname: "John",  
    lname: "Doe",  
    age: 25  
};  
let txt = "";  
for (let x in person) {  
    txt += person[x] + " "  
}
```

```
const numbers = [6, 12, 8, 11, 18];  
let txt = "";  
for (let x in numbers) {  
    txt += numbers[x] + "\t";  
}  
console.log(txt);
```

```
const numbers = [6, 12, 8, 11, 18];  
let txt = "";  
numbers.forEach(myFunction);  
function myFunction(value, index, array) {  
    txt += value + "\t";  
}  
console.log(txt);
```

2. Lập trình Javascript

»»» Vòng lặp For ... Of

- ✓ Lặp qua các **giá trị** của đối tượng
- ✓ Cú pháp:

```
for (variable of iterable) {  
    // code block to be executed  
}
```

```
const cars = ["Lexus", "Volvo", "Mini"];  
let text = "";  
for (let x of cars) {  
    text += x + "\t";  
}  
console.log(text);
```

```
let language = "JavaScript",  
    text = "";  
for (let x of language) {  
    text += x + " ";  
}  
console.log(text);
```

2. Lập trình Javascript

»»» Vòng lặp While

- ✓ Cú pháp:
**while(expression)
statements**

- ✓ Ý nghĩa:
 - B1: expression được định trị;
 - B2: nếu kết quả là true thì statement thực thi và quay lại bước 1;
 - B3: nếu kết quả là false thì thoát khỏi vòng lặp while.

→ có thể statements ko được thực hiện
lần nào cả

```
let i = 0, text = "";
while (i < 10) {
    text += "The number is " + i + "\n";
    i++;
}
console.log(text);
```

2. Lập trình Javascript

»»» Vòng lặp Do...While

- ✓ Cú pháp:

```
do {  
    statements  
} while(expression)
```

- ✓ Ý nghĩa:

- B1: statements được thực hiện;
- B2: expression được định trị:
 - nếu expression là true thì quay lại bước 1
 - nếu expression là false thì thoát khỏi vòng lặp.

→ *statements sẽ được thực hiện ít nhất 1 lần*

```
let i = 0, text = "";  
do {  
    text += "The number is " + i + "\n";  
    i++;  
}  
while (i < 10);  
console.log(text);
```

2. Lập trình Javascript

>>> Function

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

```
function name(...params) {  
    console.log(params[0]);  
    console.log(params[1]);  
    console.log(params[2]);  
    //...  
}
```

```
function name() {  
    console.log(arguments[0]);  
    console.log(arguments[1]);  
    console.log(arguments[2]);  
    //...  
}
```

```
let x = sumAll(23, 67, 271, 22, 19, 36);  
function sumAll(...numbers) {  
    let sum = 0;  
    for (let i = 0; i < numbers.length; i++) {  
        sum += numbers[i];  
    }  
    return sum;  
}  
console.log(x);
```

```
let x = sumAll(23, 67, 271, 22, 19, 36);  
function sumAll() {  
    let sum = 0;  
    for (let i = 0; i < arguments.length; i++) {  
        sum += arguments[i];  
    }  
    return sum;  
}  
console.log(x);
```

2. Lập trình Javascript

»»» Callback Function

➤ **Callback function** là hàm được truyền qua đối số khi gọi hàm khác.

```
function sayHello(params) {  
    console.log(params);  
}  
sayHello("Hello!");  
// --> Hello!
```

```
function sayHello(params) {  
    params("Welcome");  
}  
function myCallback(params) {  
    console.log(params);  
}  
sayHello(myCallback);
```

```
function sayHello(name, callback) {  
    var say = "Hi, " + name.toUpperCase();  
    return callback(say);  
}  
var result = sayHello("John", function(arg) {  
    return arg;  
});  
console.log(result);
```

```
function sayHello(params) {  
    if (typeof params === 'function') {  
        params("Welcome");  
    }  
}  
function myCallback(params) {  
    console.log(params);  
}  
sayHello(myCallback);
```

2. Lập trình Javascript

»»» Arrow Function

- Arrow function cho phép định nghĩa hàm với cú pháp ngắn gọn (được giới thiệu từ version ES6)

```
sayHello = () => {  
    return "Hello!";}      ⇔      sayHello = () => "Hello!";
```

```
sayHello = (name) => "Hi, " + name;  
console.log(sayHello("John"));
```



```
let sayHello = name => console.log("Hi, " + name);  
sayHello("John");
```

2. Lập trình Javascript

»»» Arrow Function

```
const product = {
    productCode: 1,
    productName: "Heineken",
    productPrice: '$1.2',
    getInfo: function() {
        return this.productName + " - "
        + this.productPrice;
    }
}
console.log(product.getInfo()); //Heineken - $1.2
```

```
const product = {
    productCode: 1,
    productName: "Heineken",
    productPrice: '$1.2',
    getInfo: () => {
        return this.productName + " - " +
            this.productPrice;
    }
}
console.log(product.getInfo()); //undefined - undefined
```

2. Lập trình Javascript

»»» Arrow Function

```
const Product = function(code, name, price) {
    this.code = code;
    this.name = name;
    this.price = price;
}
const heineken = new Product(1, "Heineken", "$1.2");
console.log(heineken);
```

```
Product {code: 1, name: 'Heineken',
price: '$1.2'} ⓘ
code: 1
name: "Heineken"
price: "$1.2"
► [[Prototype]]: Object
```

```
const Product = (code, name, price) => {
    this.code = code;
    this.name = name;
    this.price = price;
}
const heineken = new Product(1, "Heineken", "$1.2");
console.log(heineken);
```

✖ ► Uncaught TypeError: Product is not a constructor

2. Lập trình Javascript

»»» Một số hàm Built-in

- **eval(string):** thực thi biểu thức Javascript code/expressions dạng chuỗi

```
var x = 10;
var y = 25;
console.log("Sum: ", eval("x + y")); // Sum: 35
console.log(eval("Math.sqrt(y) * 3 + 3")); //18
```

- **isNaN:** kiểm tra đối số có phải “Not a Number”

```
let checked1 = isNaN(68),
    checked2 = isNaN("Hello");
console.log(checked1, checked2); //false true
```

2. Lập trình Javascript

»»» Một số hàm Built-in

- **Number:** chuyển giá trị đối số về dạng số

```
var x1 = true,  
    x2 = false;  
var x3 = new Date();  
var x4 = "123";  
var x5 = "123 456";  
console.log(Number(x1), Number(x2)); // 1 0  
console.log(Number(x3)); // 1635865711444  
console.log(Number(x4)); // 123  
console.log(Number(x5)); // NaN
```

- **parseInt, parseFloat:** chuyển chuỗi ký số về dạng số nguyên, số thực

```
let x = parseInt("36"),  
    y = parseFloat("39.79");  
console.log(x); // 36  
console.log(y); // 39.79
```

2. Lập trình Javascript

»»» Một số hàm Built-in

- **setTimeout**: cho phép thực thi chương trình sau 1 khoảng thời gian nhất định.

```
setTimeout(function() {
    console.log("Hello");
}, 1000);
```

- **setInterval**: cho phép thực thi chương trình lặp đi lặp lại sau 1 khoảng thời gian nhất định.

```
let secondsToCountDown = 5;
const interval = setInterval(() => {
    console.log("Seconds: ", secondsToCountDown);
    if (secondsToCountDown === 0) {
        clearInterval(interval); // time is up
    }
    secondsToCountDown--;
}, 1000);
```

2. Lập trình Javascript

»»» String

- **length:** trả về độ dài chuỗi

```
let s = "Never Say 'Never'"  
console.log(s.length); //17
```

- Các phương thức làm việc với chuỗi

indexOf	lastIndexOf	search
substring	slice	substr
replace	toUpperCase	toLowerCase
concat	split	charAt

2. Lập trình Javascript

»»» String

- **toUpperCase /toLowerCase** : chuyển chuỗi về in hoa / thường

```
let s = "Never Say 'Never'"  
console.log(s.toUpperCase()); // NEVER SAY 'NEVER'  
console.log(s.toLowerCase()); // never say 'never'
```

- **trim**: xóa khoảng trắng thừa ở hai đầu chuỗi.

```
let s = "      Never Say 'Never'      ";  
console.log(s.trim());
```

2. Lập trình Javascript

»»» String

- **indexOf(searchString, position)**: trả về vị trí đầu tiên tìm thấy chuỗi con trong chuỗi gốc, trả về -1 nếu không tìm thấy.

```
let s = "Never Say 'Never'"  
console.log(s.indexOf("Never")); // 0  
console.log(s.indexOf("Never", 2)); // 11
```

- **lastIndexOf**: trả về vị trí cuối cùng tìm thấy chuỗi con trong chuỗi gốc, trả về -1 nếu không tìm thấy.

```
let s = "Never Say 'Never'"  
console.log(s.lastIndexOf("Never")); // 11
```

2. Lập trình Javascript

»»» String

- **search:** trả về vị trí đầu tiên tìm thấy chuỗi chon trong chuỗi gốc, trả về -1 nếu không tìm thấy, cho phép tìm kiếm theo regular expression.

```
let s = "Never Say 'Never'"  
console.log(s.search("Never")); // 0  
//Tìm kiếm theo regular expression  
//vd: tìm kiếm không phân biệt chữ hoa, chữ thường  
console.log(s.search(/never/i)); // 0
```

- **charAt(index):** trả về ký tự tại vị trí theo chỉ mục index.

```
let beer = 'Tiger';  
console.log(beer.charAt(2)); // g  
console.log(beer[3]); // e
```

2. Lập trình Javascript

»»» String

- **substring(start, end)**: trích lọc chuỗi con trong chuỗi gốc từ vị trí **start** đến **< end**.

```
let s = "Never Say 'Never'"  
console.log(s.substring(6, 9)); // Say  
console.log(s.substring(6)); // Say 'Never'
```

- **slice(start, end)**: tương tự như hàm substring nhưng cho phép truyền số âm tính từ cuối chuỗi.

```
let s = "Never Say 'Never'"  
console.log(s.slice(6, 9)); // Say  
console.log(s.slice(-6, -1)); // Never  
console.log(s.slice(-7)); // 'Never'
```

- **substr(from, length)**: trích lọc chuỗi con trong chuỗi gốc tính từ vị trí **from** lấy **length** ký tự.

```
let s = "Never Say 'Never'"  
console.log(s.substr(6, 3)); // Say  
console.log(s.substr(10)); // 'Never'
```

2. Lập trình Javascript

»»» String

- **replace:** thay thế chuỗi.

```
let s = "Welcome to UEL, UEL"
console.log(s.replace("UEL", "FIS")); // Welcome to FIS, UEL
//Thay thế tất cả
console.log(s.replace(/UEL/g, "FIS")); // Welcome to FIS, FIS
//Thay thế tất cả, không phân biệt chữ hoa, chữ thường
console.log(s.replace(/Uel/gi, "FIS")); // Welcome to FIS, FIS
```

2. Lập trình Javascript

»»» String

- concat: nối chuỗi.

```
let s1 = "Welcome",
    s2 = "to",
    s3 = 'Vietnam';
console.log(s1 + " " + s2 + " " + s3);
console.log(s1.concat(" ", s2, " ", s3));
```

- split: tách chuỗi thành mảng theo chuỗi phân tách chỉ định.

```
let beers = 'Heineken, Tiger, Sapporo';
console.log(beers.split(', '));
//['Heineken', 'Tiger', 'Sapporo']
```

```
let beer = 'Tiger';
console.log(beer.split(''));
//['T', 'i', 'g', 'e', 'r']
```

2. Lập trình Javascript

>>> Array

- **Array (Mảng):** cho phép lưu nhiều giá trị phần tử (có thể không cùng kiểu dữ liệu) trong cùng 1 biến.

```
var beer1 = "Heineken";
var beer2 = "Tiger";
var beer3 = "Sapporo";
```



```
var beers = ["Heineken", "Tiger",
"Sapporo"];
var beers = new Array("Heineken",
"Tiger", "Sapporo");
```

Kiểm tra kiểu dữ liệu:

```
console.log(typeof beers); // object
console.log(typeof {}); // object
console.log(Array.isArray(beers)); // true
console.log(beers instanceof Array); // true
console.log(Array.isArray({})); // false
console.log({} instanceof Array); // false
```

2. Lập trình Javascript

»»» Array

- **toString()**: chuyển array thành chuỗi, các phần tử cách nhau bởi dấu “,”

```
const beers = ["Heineken", "Tiger",
  "Sapporo"];
console.log(beers.toString());
//--> Heineken,Tiger,Sapporo
```

- **join()**: nối các phần tử của array thành chuỗi, các phần tử cách nhau bởi chuỗi đối số của hàm.

```
const beers = ["Heineken", "Tiger",
  "Sapporo"];
console.log(beers.join(" - "));
//--> Heineken - Tiger - Sapporo
```

2. Lập trình Javascript

»»» Array

- **pop()**: xóa phần tử cuối của array và trả về phần tử đã xóa.

```
const beers = ["Heineken", "Tiger", "Sapporo"];
console.log(beers.pop()); // Sapporo
console.log(beers); // ['Heineken', 'Tiger']
```

- **shift()**: xóa phần tử đầu của array và trả về phần tử đã xóa.

```
const beers = ["Heineken", "Tiger", "Sapporo"];
console.log(beers.shift()); // Heineken
console.log(beers); // ['Tiger', 'Sapporo']
```

2. Lập trình Javascript

»»» Array

- **push()**: thêm các phần tử vào cuối array và trả về số lượng phần tử của array.

```
const beers = ["Heineken", "Tiger", "Sapporo"];
console.log(beers.push("Corona Extra")); // 4
console.log(beers);
// ['Heineken', 'Tiger', 'Sapporo', 'Corona Extra']
```

- **unshift()**: thêm các phần tử vào đầu array và trả về số lượng phần tử của array.

```
const beers = ["Heineken", "Tiger", "Sapporo"];
console.log(beers.unshift("Corona Extra")); // 4
console.log(beers);
// ['Corona Extra', 'Heineken', 'Tiger', 'Sapporo']
```

2. Lập trình Javascript

»»» Array

- `splice(start, deleteCount, ...items)`: xóa, thêm phần tử tại vị trí bất kỳ trong array.

```
const beers = ["Heineken", "Tiger", "Sapporo"];
console.log(beers.splice(1, 1)); // "Tiger"
console.log(beers); // ['Heineken', 'Sapporo']
```

```
const beers = ["Heineken", "Tiger", "Sapporo"];
console.log(beers.splice(1, 1, "Corona")); // "Tiger"
console.log(beers); // ['Heineken', 'Corona', 'Sapporo']
```

```
const beers = ["Heineken", "Tiger", "Sapporo"];
console.log(beers.splice(1, 0, "Corona", "Strongbow")); // []
console.log(beers);
// ['Heineken', 'Corona', 'Strongbow', 'Tiger', 'Sapporo']
```

2. Lập trình Javascript

»»» Array

- concat(): nối các array.

```
const beerList1 = ["Heineken", "Tiger", "Sapporo"];
const beerList2 = ["Corona", "Strongbow"];
console.log(beerList1.concat(beerList2));
// ['Heineken', 'Tiger', 'Sapporo', 'Corona', 'Strongbow']
```

- slice(start, end): cắt array từ vị trí start đến < end.

```
const beerList = ["Heineken", "Tiger", "Sapporo"];
console.log(beerList.slice(1, 2)); // ['Tiger']
console.log(beerList.slice(-2, -1)); // ['Tiger']
console.log(beerList.slice(-1)); // // ['Sapporo']
```

2. Lập trình Javascript

»»» Array

- `sort()`, `reverse()`: sắp xếp mảng.

```
const beerList = ["Heineken", "Tiger", "Sapporo"];
beerList.sort();
console.log(beerList); //['Heineken', 'Sapporo', 'Tiger']
beerList.reverse();
console.log(beerList); //['Tiger', 'Sapporo', 'Heineken']
```

```
const marks = [4, 2, 10, 8, 15];
marks.sort();
console.log(marks); // [10, 15, 2, 4, 8]
```

```
const marks = [4, 2, 10, 8, 15];
marks.sort(function(a, b) {
    return a - b;
});
console.log(marks); // [2, 4, 8, 10, 15]
```

2. Lập trình Javascript

»»» Array

- **forEach()**: duyệt qua các phần tử mảng.

```
let numbers = [36, 6, 9, 18, 22];
numbers.forEach(myFunction);
function myFunction(value, index, array) {
  console.log(index + ":" + value);
}
```

- **map()**: tạo ra mảng mới bằng cách thực hiện logic bất kỳ trên mỗi phần tử mảng ban đầu

```
let arr1 = [36, 6, 9, 18, 22];
let arr2 = arr1.map(myFunction);
function myFunction(value, index, array) {
  return value ** 2;
}
console.log(arr2); // [1296, 36, 81, 324, 484]
```

2. Lập trình Javascript

>>> Array

- **reduce()**: thực hiện logic tính toán trên các phần tử mảng và trả về giá trị duy nhất.

```
let numbers = [3, 2, 4, 1, 6, 8];
let sum = numbers.reduce(myFunction);
function myFunction(total, value, index, array) {
    return total + value;
}
console.log(sum); // 24
```

```
let numbers = [3, 2, 4, 1, 6, 8];
let sum = numbers.reduce(function(total, value) {
    return total + value;
}, 0);
console.log(sum); // 24
```

```
let numbers = [3, 2, 4, 1, 6, 8];
let sum = numbers.reduce((total, value) =>
(total + value), 0);
console.log(sum); // 24
```

```
let numbers = [3, 2, 4, 1, 6, 8];
let sum = numbers.reduce(myFunction, 0);
function myFunction(total, value, index, array) {
    return total + value;
}
console.log(sum); // 24
```

Giá trị khởi tạo cho
biến tích trữ

- **reduceRight()**: tương tự như **reduce()** nhưng duyệt mảng từ phải qua trái.

2. Lập trình Javascript

»»» Array

- **every()**: kiểm tra tất cả phần tử của mảng thỏa điều kiện chỉ định.

```
const numbers = [36, 3, 11, 16, 19, 22];
const over20 = numbers.every(myFunction);
function myFunction(value, index, array) {
    return value > 20;
}
console.log(over20); // false
```

- **some()**: kiểm tra **một** vài phần tử của mảng thỏa điều kiện chỉ định.

```
const numbers = [36, 3, 11, 16, 19, 22];
const over20 = numbers.some(myFunction);
function myFunction(value, index, array) {
    return value > 20;
}
console.log(over20); // true
```

2. Lập trình Javascript

»»» Array

- **indexOf()**: trả về vị trí của phần tử đầu tiên tìm thấy trong mảng, nếu không tìm thấy trả về -1.

```
let numbers = [3, 2, 4, 3, 2, 8];
let position = numbers.indexOf(2);
console.log(position); // 1
```

- **lastIndexOf()**: trả về vị trí của phần tử cuối cùng tìm thấy trong mảng, nếu không tìm thấy trả về -1.

```
let numbers = [3, 2, 4, 3, 2, 8];
let position = numbers.lastIndexOf(2);
console.log(position); // 4
```

2. Lập trình Javascript

»»» Array

- **find()**: trả về phần tử đầu tiên tìm thấy trong mảng, nếu không tìm thấy trả về *undefined*.

```
let beers = ["Heineken", "Tiger", "Sapporo"];
let beer = beers.find(function(value) {
  return value == "Tiger";
});
console.log(beer); // Tiger
```

- **filter()**: tạo ra mảng mới với **các** phần tử của mảng thỏa mãn điều kiện chỉ định.

```
const numbers = [36, 3, 11, 16, 19, 22];
const over20 = numbers.filter(myFunction);
function myFunction(value, index, array) {
  return value > 20;
}
console.log(over20); // [36, 22]
```

2. Lập trình Javascript

»»» Array

- **includes()**: kiểm tra phần tử có tồn tại trong mảng, chuỗi con trong chuỗi gốc.

```
let beers = ["Heineken", "Tiger", "Sapporo"];
console.log(beers.includes("Tiger")); // true
```

```
let s = "No Pain No Gain!"
console.log(s.includes("No", 3)); //true
```

- **findIndex()**: trả về chỉ mục index của phần tử đầu tiên thỏa điều kiện chỉ định.

```
const numbers = [1, 3, 11, 16, 19, 22];
const over15 = numbers.findIndex(myFunction);
function myFunction(value, index, array) {
  return value > 15;
}
console.log(over15); // 3
```

2. Lập trình Javascript

»»» Array

- **from()**: returns an Array object from any object with a length property or any iterable object.

```
let s = "Hello!";
let arr = Array.from(s);
console.log(arr); //['H', 'e', 'l', 'l', 'o', '!']
```

- **key()**: returns an Array Iterator object with the keys of an array.

```
let beers = ["Heineken", "Tiger", "Sapporo"];
for (let key of beers.keys()) {
    console.log(key);
}
```

//0
//1
//2

2. Lập trình Javascript

»» Object

Object (đối tượng): giống như một đối tượng cụ thể trong thế giới thực.

Mỗi đối tượng có các *thuộc tính (đặc điểm)* và các *phương thức (hành động / hành vi)* riêng.

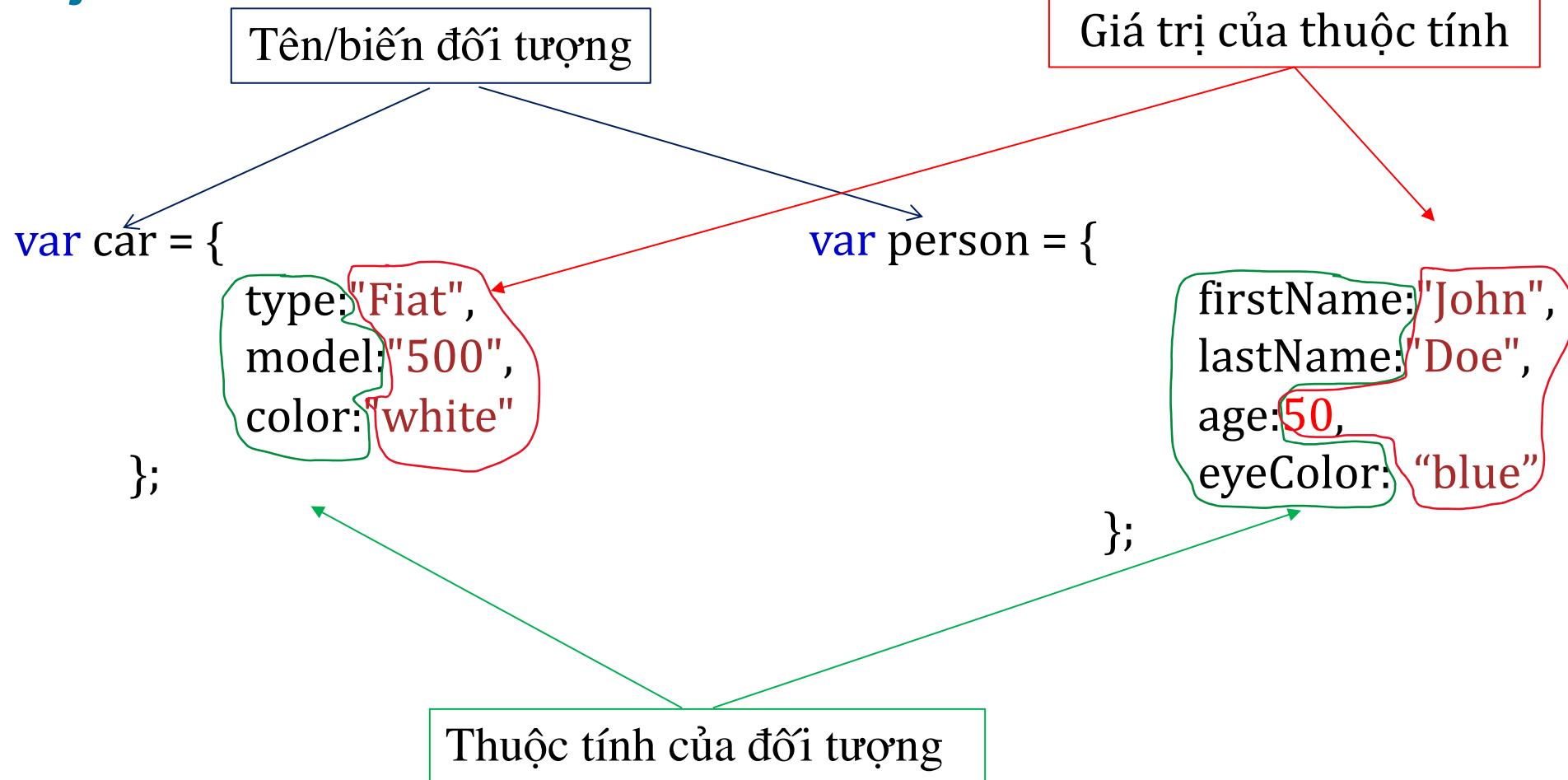
- Thuộc tính (attribute): mô tả đặc điểm của đối tượng.
 - Phương thức (method): mô tả hoạt động của đối tượng.
- ➔ Những *đối tượng giống nhau* sẽ được xếp vào cùng một loại hay cùng một *lớp* (class).



- Truy xuất thuộc tính:
`objectName.propertyName` hoặc
`objectName["propertyName"]`
- Truy xuất phương thức:
`objectName.methodName()`

2. Lập trình Javascript

»» Object



2. Lập trình Javascript

»»» Object

```
const Product = function(code, name, price) {  
    this.code = code;  
    this.name = name;  
    this.price = price;  
};  
let p = new Product(1, "Heineken", 19000);  
console.log(p);  
/* Product {  
    code: 1,  
    name: 'Heineken',  
    price: 19000  
}*/
```

```
class Employee {  
    constructor(id, name, dob) {  
        this.id = id;  
        this.name = name;  
        this.dob = dob;  
    }  
    getInfo() {  
        return this.id + ": " + this.name;  
    }  
}  
let e = new Employee(1, "Peter", new Date  
('2000-04-18'));  
console.log(e);  
/*Employee {id: 1, name: 'Peter', dob: Tue Apr  
18 2000 07:00:00 GMT+0700 (Indochina Time)}*/
```

2. Lập trình Javascript

»»» Date Object

- **Date:** đối tượng thể hiện thời gian (ngày, tháng, năm, giờ, phút, giây, mili giây)

```
new Date()
new Date(year, month, day, hours, minutes,
seconds, milliseconds)
new Date(milliseconds)
new Date('date string')
```

```
console.log(new Date()); //Current time
console.log(new Date(2021, 10, 06));
/*Sat Nov 06 2021 00:00:00 GMT+0700
(Indochina Time)
January = 0 ... December = 11*/
console.log(new Date("2021-11-06"));
//Sat Nov 06 2021 07:00:00 GMT+0700
```

2. Lập trình Javascript

»» Date Object

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

2. Lập trình Javascript

>>> Date Object

Method	Description
getFullYear()	Get the year as a four digit number (yyyy)
getMonth()	Get the month as a number (0-11)
getDate()	Get the day as a number (1-31)
getHours()	Get the hour (0-23)
getMinutes()	Get the minute (0-59)
getSeconds()	Get the second (0-59)
getMilliseconds()	Get the millisecond (0-999)
getTime()	Get the time (milliseconds since January 1, 1970)
getDay()	Get the weekday as a number (0-6)
Date.now()	Get the time. ECMAScript 5.

2. Lập trình Javascript

»»» Math Object

- **Math:** cung cấp một số phương thức xử lý logic toán học

```
Math.PI; // returns 3.141592653589793
```

```
Math.round(4.7); // returns 5
```

```
Math.round(4.4); // returns 4
```

```
Math.pow(8, 2); // returns 64
```

```
Math.sqrt(64); // returns 8
```

```
Math.abs(-4.7); // returns 4.7
```

```
Math.ceil(4.4); // returns 5
```

```
Math.floor(4.7); // returns 4
```

```
Math.min(0, 150, 30, 20, -8, -200); // returns -200
```

```
Math.max(0, 150, 30, 20, -8, -200); // returns 150
```

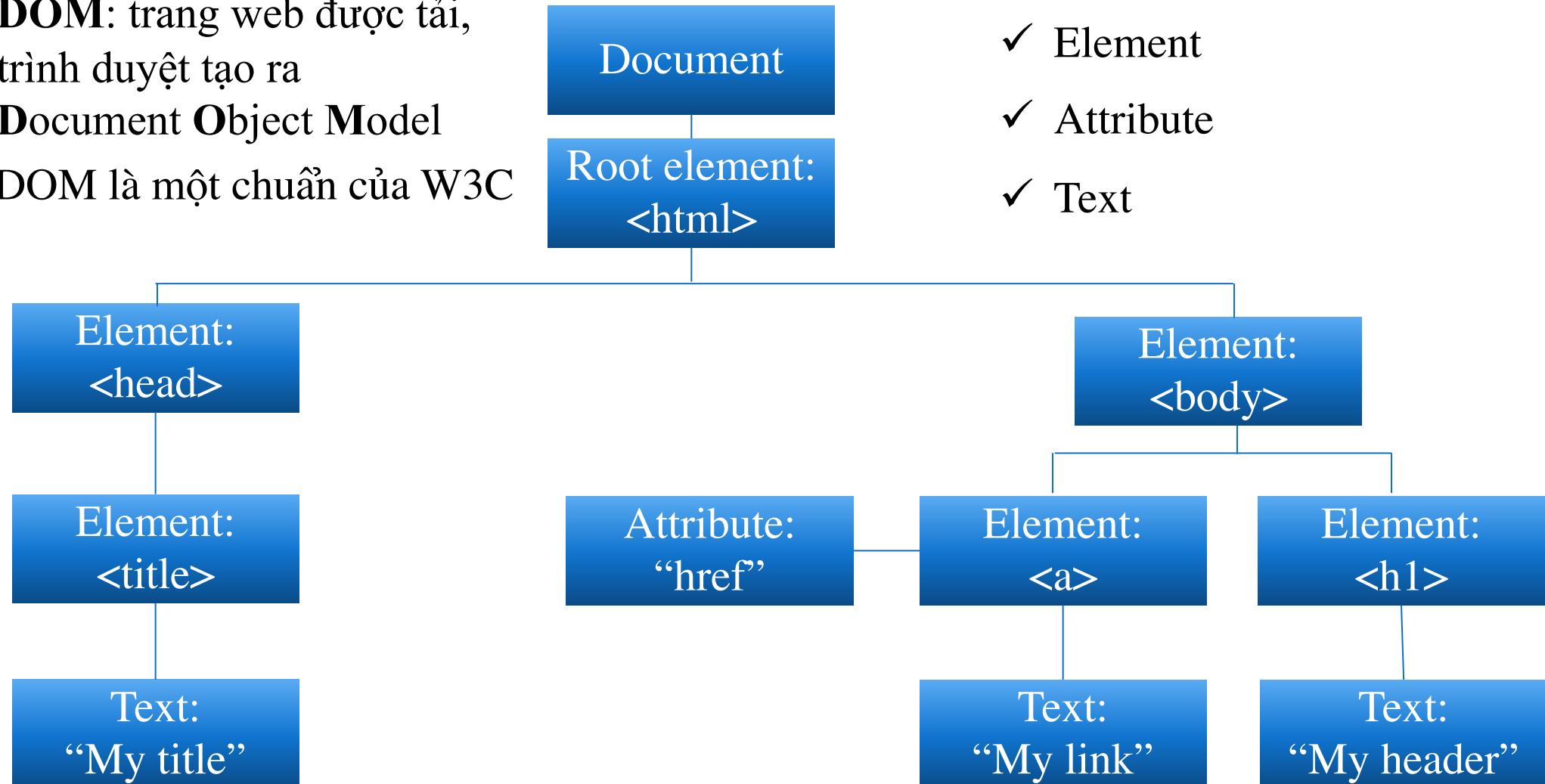
```
Math.sin(90 * Math.PI / 180); // returns 1 (the sine of 90 degrees)
```

```
Math.cos(0 * Math.PI / 180); // returns 1 (the cos of 0 degrees)
```

....

3. Cấu trúc Document Object Model (DOM)

- **DOM:** trang web được tải, trình duyệt tạo ra
Document Object Model
DOM là một chuẩn của W3C



3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM

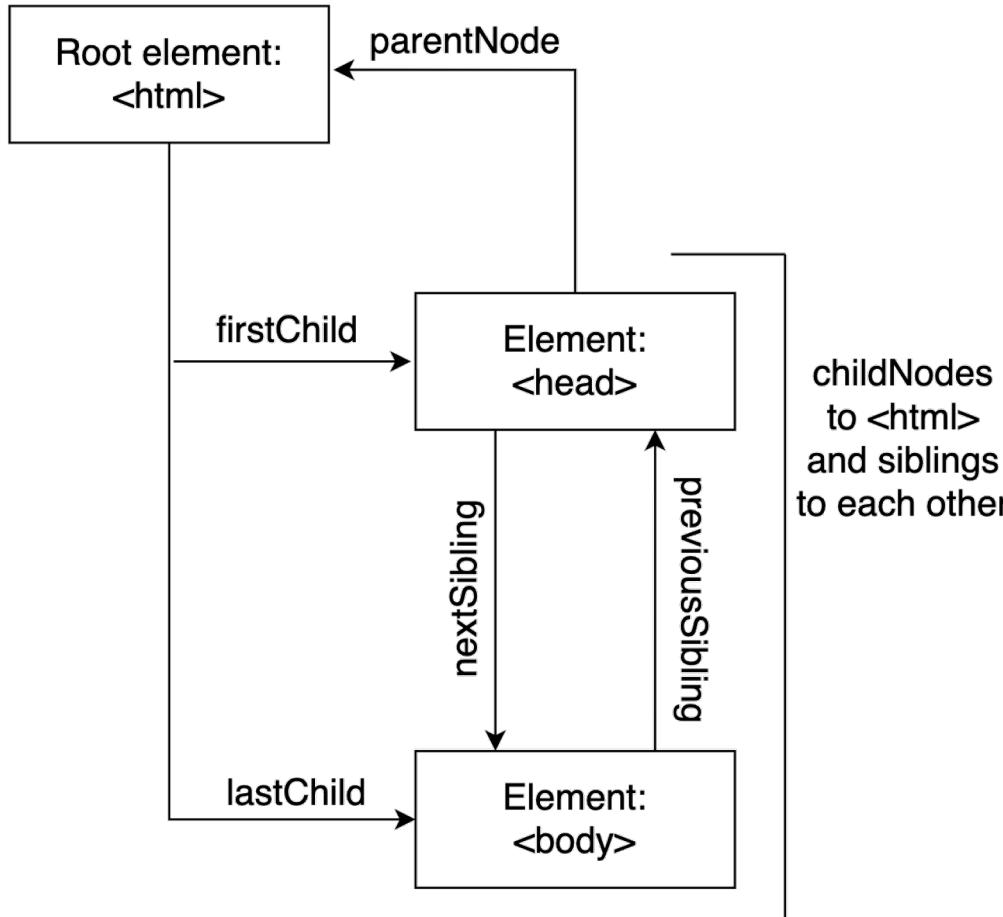
```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

- Node gốc là <html>
- Tất cả các node còn lại trong DOM chứa trong node <html>
- Node <html> có hai node con là <head> và <body>
- Node <head> chứa node <title>
- Node <body> chứa node <h1>, <p>

3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```



3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM Methods / Attributes

`document.createElement("tag")`: tạo một element

`document.getElementById("id")`: tìm một element theo Id

`document.getElementsByTagName("tag")`: tìm các element theo tag

`document.getElementsByClassName("class")`: tìm các element theo class

3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM Methods / Attributes

✓ Một số thuộc tính của Element:

- **x.id** – id của x
- **x.tagName** – tên tag của x
- **x.innerHTML** – giá trị text của x
- **x.parentNode** – nút cha của x
- **x.childNodes** – các nút con của x
- **x.attributes** – các thuộc tính của x
- **x.className** – tên class của x

✓ Một số phương thức của Element:

- **x.getElementsByClassName("class")**: tìm các element theo class
- **x.getElementsByTagName("name")**: tìm các element theo tên tag
- **x.querySelector("selectors")**: tìm các element theo css selector
- **x.appendChild(node)**: thêm nút con đến x
- **x.removeChild(node)**: xóa nút con từ x
- **x.setAttribute("attr", "value")**: đặt lại giá trị attribute cho x

3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM Methods / Attributes

```
<body>
    <p id="intro">Hello World!</p>
    <script type="text/javascript">
        let content = document.getElementById("intro").
        ➔ innerHTML;
        document.write("<p>The text from the intro
            paragraph: " + content + "</p>");
    </script>
</body>
```

```
<body>
    <p id="intro">Hello World!</p>
    <script type="text/javascript">
        let content = document.getElementById("intro").
        ➔ childNodes[0].nodeValue;
        document.write("<p>The text from the intro
            paragraph: " + content + "</p>");
    </script>
</body>
```

3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM Methods / Attributes

```
<body>
    <p id="intro">Welcome to Vietnam</p>
    <script type="text/javascript">
        let element = document.getElementById("intro")
        document.write("<p>The text from the intro
        paragraph: " + element.innerHTML + "</p>");
    </script>
</body>
```



3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM Methods / Attributes

```
<body>
    <p>Hello World!</p>
    <p>The DOM is very useful!</p>
    <p>This example demonstrates the <b>getElementsByTagName</b>
method.</p>
    <script type="text/javascript">
        p_elements = document.getElementsByTagName("p"); //lấy
        tất cả thẻ có tên là p
        document.write("Text of first paragraph: " + p_elements
[0].innerHTML); //ghi ra giá trị của thẻ đầu tiên
    </script>
</body>
```

3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM Methods / Attributes

```
<body>
    <p>Hello World!</p>
    <p>The DOM is very useful!</p>
    <p>This example demonstrates the <b>getElementsByTagName</b>
method.</p>
    <script type="text/javascript">
        p_elements = document.getElementsByTagName("p");
        for (i = 0; i < p_elements.length; i++) {
            document.write(p_elements[i].innerHTML + "<br/>");
        }
    </script>
</body>
```

3. Cấu trúc Document Object Model (DOM)

➤ HTML DOM Events

```
<body>
    <p id="para">Hello world!</p>
    <input type="button" onclick="changeText()"
    value="Change text" />
    <script type="text/javascript">
        function changeText() {
            document.getElementById("para").innerHTML =
            "New text!";
        }
    </script>
</body>
```

```
<input type="button" onclick="document.body.style.
backgroundColor='lavender';" value="Change background
color" />
```

Xem thêm: https://www.w3schools.com/jsref/dom_obj_event.asp

3. Cấu trúc Document Object Model (DOM)

- Tham khảo thêm về HTML DOM:

https://www.w3schools.com/jsref/dom_obj_document.asp

https://www.w3schools.com/jsref/dom_obj_all.asp

https://www.w3schools.com/jsref/dom_obj_attributes.asp

https://www.w3schools.com/jsref/dom_obj_event.asp

https://www.w3schools.com/jsref/dom_obj_style.asp

