JAVASCRIPT (Website)

AUGUST 13

Tự học JavaScript

Authored by: Tạ Quang Tùng

TUVO SHARK

MỤC LỤC

BÀI 1: GIỚI THIỆU VỀ JAVASCRIPT

1. JavaScript với HTML và CSS:

- JavaScript là ngôn ngữ lập trình có khả năng làm thay đổi cấu trúc HTML và thuộc tính của style CSS bằng việc xác thực dữ liệu JS
- Sử dụng thẻ <script> để nhúng mã JavaScript vào trong file HTML: (file JavaScript luôn ở vị trí cuối cùng)
    - Cách 1: Viết độc lập trong thẻ <script>:

```
<script>
        alter(“Hello World”);
</script>
```

```
console.log('Hello, World!');
document.write('Hello, World!');
```

    - Cách 2: Viết trực tiếp:

```
<script>
        document.getElementById("demo").innerHTML = "Content JS";
        document.getElementById(“demo”).innerText = “Content Print”;
</script>
```

→ Đặt id = “demo” từ HTML

- innerHTML → thay giá trị từ HTML
- innerText → thay bằng đoạn text

    - Cách 3: Xử dụng External JavaScript: (độc lập với code của HTML)

```
<!DOCTYPE html>
<html>
<body>
        <script src="myJavaScript.js"> </script>
</body>
</html>
```

→ Chuyển các trang web qua lại với nhau

2. Chú thích JavaScript: (giống Java)

- // Chú thích một dòng
- Chú thích nhiều dòng:

```
/*
   Đây là chú thích nhiều dòng
*/
```

3. Các Keywords cần nhớ trong JavaScript:

Keyword	Mô tả
break	Thoát lập tức ra khỏi vòng lặp và dừng chương trình vòng lặp
continue	Nhảy ra khỏi vòng lặp và tiếp tục chạy
do ... while	Vòng lặp thực hiện câu lệnh trước rồi mới kiểm tra
for	Vòng lặp for thực hiện câu lệnh lặp phía trong nó
function - return	Gọi hàm dựng và trả kết quả của Function
if ... else	Câu lệnh điều kiện Nếu ... thì ...
switch	Kiểm tra biến tùy vào từng giá trị cụ thể
try ... catch	Câu lệnh chạy trong try, nếu bị lỗi thì sang catch

4. Tương tác với alter, prompt, confirm khi dùng với HTML:

- alter(message): dùng để hiển thị hộp thoại, bao gồm nội dung tin nhắn và nút bấm (button) OK. Khi người dùng bấm vào OK thì hộp thoại sẽ đóng lại.
- prompt(title, "giá trị mặc định trong ô input"): cho phép người dùng nhập vào string
- confirm(message): đưa ra thông báo xác nhận bằng cách chọn OK hoặc Cancel

→ Nhập xuất dữ liệu đầu vào trong JavaScript khi chỉ được gọi trong HTML phần body

+ B0: Tạo thẻ ngoài <script> có chưa ID

+ B1: Nhập dữ liệu đầu vào trong thẻ <script>
let nameBien = prompt("Nhập input: ");

+ B2: Xuất dữ liệu ra màn hình trong thẻ <script>
alter(nameBien);

+ B3: Liên kết với id trên HTML:
const ketQua = document.getElementById(nameID);
ketQua.innerHTML = output;

BÀI 2: BIẾN VÀ KIỂU DỮ LIỆU

1. Khai báo biến:

- Có ba cách khai báo biến:
  - Biến toàn cục Global → var
  - Biến cục bộ Local → let
  - Hằng số → const
- Công thức:
  - L1 (không định nghĩa): var nameVariable; let nameVariable;
  - L2 (có định nghĩa): var nameVariable = <value>; let nameVariable;
  - L3: nameVariable = <value>;
  - L4: (đối với nhiều biến):
    - var nameVariable1 = <value1>, nameVariable2 = <value2>;
    - let nameVariable1 = <value1>, nameVariable2 = <value2>;
  - L5 (biến HẰNG): const nameVariable = <constValue>;

2. Các kiểu dữ liệu:

- Kiểu số (Number) → var nameNumber =30;
- Kiểu chuỗi (String) → var nameString = “Chuỗi”;
- Kiểu mảng (Array) → var nameArray = [val1, val2, val3];
- Kiểu đối tượng (Object) → var nameObject = {firstNameL: “Tạ”, lastName: “Tùng”}

3. Kiểu dữ liệu Boolean():

- Biến mặc định là đúng: var nameVariable = true;
- Biến mặc định là sai: var nameVariable = false;
  → Riêng với Boolean() ta có thể chuyển đổi qua lại giữa các đối tượng String() hoặc Number()

BÀI 3: CÁC LOẠI TOÁN TỬ

1. Toán tử số học:

Phép toán	Mô tả	Ví dụ
+	Phép cộng	var tong = x + y;
-	Phép trừ	var hieu = x – y;
*	Phép nhân	var nhan = x * y;
/	Phép chia	var chia = x / y
%	Phép chia lấy phần dư	var du = x % y;
++	Cộng dồn +1	x++
--	Trừ dồn -1	x--

2. Toán tử gán:

Phép toán	Mô tả	Cụ thể
=	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

3. Toán tử so sánh:

Phép toán	Mô tả
==	So sánh ngang bằng với giá trị
===	So sánh ngang bằng với cả giá trị và kiểu dữ liệu
!=	So sánh hai vế có giá trị khác nhau
!==	So sánh hai vế có giá trị hoặc kiểu khác nhau
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
!	Phép phủ định

4. Toán tử Logic:

Phép toán	Mô tả	Cụ thể
|| (Hoặc)	a || b	AND
&& (Và)	a && b	OR

5. Loại phép toán:

- **typeof** → Trả về kiểu dữ liệu của biến
- **instanceof** → Kiểm tra đối tượng này có thuộc kiểu dữ liệu đối tượng kia không

BÀI 4: ĐỐI TƯỢNG CHUỖI STRING

1. Thuộc tính String:

Property	Mô tả
Constructor	Trả về một tham chiếu tới hàm String mà tạo đối tượng đó
Length	Trả về độ dài chuỗi
Prototype	Cho phép thêm các thuộc tính và phương thức tới một đối tượng

2. Phương thức:

- **"string".charAt(index)** → trả về chỉ số index của ký tự trong chuỗi

```
var str = "HELLO WORLD";
str.charAt(0);
→ H
```

- **"string".concat(" ", str1, str2, ...)** → cộng gộp văn bản từ các chuỗi

```
var text1 = "Hello";
var text2 = "World";
text3 = text1.concat(" ",text2);
→ Hello Word
```

- **"string".indexOf(strg)** → trả về vị trí xuất hiện của strg, nếu không có in -1

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
→ 7
```

- **"string".lastIndexOf(strg)** → trả về vị trí xuất hiện cuối cùng của strg, nếu không in -1

```
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
→ 21
```

- **"string".replace(chuỗi muốn thay, từ mới)** → thay thế từ trong một chuỗi

```
str = "Please visit Microsoft!";
var n = str.replace("Microsoft","W3Schools");
→ Please visit W3Schools!
```

- **"string".search(str)** → tìm kiếm xem str có xuất hiện trong string không, nếu có hiển thị chỉ số trong chuỗi đó

```
var str = "Please locate where 'locate' occurs!";
var pos = str.search("locate");
→ 7
```

- **“string”.slice(begin, end)** → cắt lọc chuỗi từ chỉ số begin tới end
- **“string”.substring(start, end)** → cắt lọc chuỗi từ begin tới end

var str = "Apple, Banana, Kiwi"; **var res = str.slice(7,13);** **var res = str.substring(7,13);**	**→ Banana**

- **“string”.substr(start, length)** → Trả về ký tự của chuỗi từ start có độ dài length

var str = "Apple, Banana, Kiwi"; **var res = str.substr(7,6);** **→ Banana**

3. Một số các phương thức khác:

- **Mã Code và String:**
  - **“string”.charCodeAt(index)** → trả về giá trị Unicode trong bảng mã của ký tự
  - **“string”.fromCharCode()** → chuyển đổi giá trị Unicode thành ký tự
- **Đối tượng và String:**
  - **“string”.valueOf()** → trả về giá trị gốc của đối tượng đã xác định (Object → Value)
  - **“string”.localeCompare(stringNew)** → So sánh hai chuỗi khớp nhau như nào (Object == Object)
- **Văn bản và String:**
  - **“string”.trim()** → loại bỏ khoảng trắng dư thừa của chuỗi
  - **“string”.split(“ “, limitNumber)** → Tách chuỗi thành các từ
  - **“string”.toLocaleLowerCase()** → chuyển tất cả về dạng chữ viết thường (Object → object)
  - **“string”.toLocaleUpperCase()** → chuyển tất cả về dạng chữ viết HOA (Object)
  - **“string”.toLowerCase()** → chuyển tất cả về dạng chữ viết thường (String)
  - **“string”.toUpperCase()** → chuyển tất cả về dạng chữ viết HOA (String)
  - **“string”.toString()** → trả về chuỗi biểu diễn đối tượng đã xác định (Object)

4. Khởi tạo đối tượng String:

- ***var stringObject = new String(string);***

BÀI 5: ĐỐI TƯỢNG SỐ NUMBER

1. Phương thức Global Number:

Method	Mô tả	Ví dụ
Number()	Trả về một số từ đối số được truyền vào (lớn hơn parseFloat và parseInt)	**x = true, y = "10 20"; → Numer(x) = 1 → Number(y) = NaN**
parseFloat()	Trả về số thực từ đối số truyền vào	**parseFloat("10.22") = 10.22**
parseInt()	Trả về số nguyên từ đối số truyền vào	**parseInt("10") = 10**
toString()	Trả về số dạng chuỗi	**x.toString() = "x"**
valueOf()	Chuyển Object thành Value của nó	**x.valueOf() = x**
toExponential()	Trả về chuỗi với một số được làm tròn từ hàm mũ	**num.toExponential()**
toFixed()	Trả về chuỗi với số được làm tròn	**Var num = 5.56 num.toFixed() = 6**
toLocaleString()	Trả về phiên bản giá trị chuỗi của số hiện tại	
toPrecision()	Định nghĩa bao nhiêu chữ số (gồm cả phần nguyên và phần thập phân) để hiển thị một số	

2. Các quy ước số Number:

- Number.MAX_VALUE → Số lớn nhất có thể trong JavaScript
- Number.MIN_VALUE → Số bé nhất có thể trong JavaScript
- NaN → Đại diện giá trị không phải Number
- Number.NEGATIVE_INFINITY → Số âm xác định tối đa trong JavaScript (âm vô cực)
- Number.POSITIVE_INFINITY → Số dương xác định tối đa trong JavaScript (dương vô cực)
- prototype → Thuộc tính tĩnh của đối tượng Number để gán các phương thức mới tới đối tượng Number hiện tại
- constructor → Hàm trả về sự minh họa đối tượng Number này

3. Khởi tạo đối tượng Number:

- ***var numberObject = new Number(number);***

BÀI 6: CÂU LỆNH ĐIỀU KIỆN

1. Câu lệnh if – else if - else:

- Cú pháp đầy đủ:

```
if (condition1) {
        // Thực thi câu lệnh khi condition1 đúng
} else if (condition2) {
        // Thực thi câu lệnh khi condition2 đúng
} else if (condition3) {
        ...
} else {
        // Thực thi câu lệnh này khi tất cả các điều kiện trên sai
}
```

- Cú pháp lồng nhau:

```
if (condition1) {
        // Thực thi câu lệnh khi condition1 đúng
        if (condition2) {
                // Thực thi khi câu lệnh condition2 đúng
        }
        ...
} else {
        block of code to be executed if the condition is false
}
```

2. Toán tử ? và ??:

(condition) ? <true> : <false>;	**a ?? b ⇔ x = (a !== null && a !== undefined) ? a : b;** + Chọn a nếu nó không null hoặc undefined + Chọn b trong trường hợp ngược lại

3. Câu lệnh switch – case - default:

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

- Câu lệnh switch-case được tính toán một lần dựa vào điều kiện case cụ thể
- Nếu có kết quả phù hợp sẽ kết hợp thêm câu lệnh break để thoát khỏi switch – case

4. Câu lệnh try – catch – finally:

```
try {
        // Chạy câu lệnh code ở đâu
} catch(err) {
       // nếu có lỗi này thì nhảy sang đây
       // err is the error object
} finally {
      // Nhảy sang khối lệnh này khi after try/catch
}
```

BÀI 7: VÒNG LẶP

1. Vòng lặp For:

For duyệt Number	For duyệt Array

for (start; end; step) {
// Thực thi khối lệnh lặp
}
	var array;
	for (element in array) {
	// Lấy từng phần tử trong mảng
	}

for (i = 0; i < 5; i++) {
"The number is " + i;
}
	var person = {fname:"John", lname:"Doe", age:25};
	var text = "";
	var x;
	for (x in person) {
	text += person[x];
	}

2. Vòng lặp While:

```
while (condition) {
    // Thực hiện câu lệnh lặp
{
```

3. Vòng lặp Do – While:

```
do {
    // Thực hiện lặp trước khi kiểm tra
} while (condition);
```

```
do {
        text += "The number is " + i;
        i++;
} while (i < 10);
```

4. Câu lệnh Break và Continue:

- Câu lệnh "**break**" thoát lập tức ra khỏi vòng lặp
- Câu lệnh "**continue**" sẽ bỏ qua lần lặp hiện tại nhưng vẫn tiếp tục thực hiện những lần tiếp theo khác

  → Thường hay được kết hợp với câu lệnh điều kiện và vòng lặp để đưa ra điều kiện thích hợp

BÀI 8: MẢNG DỮ LIỆU ARRAY

1. Khởi tạo mảng Array:

- **Cách 1:** (Gọi mảng trực tiếp)

```
var arrayName = [item1, item2, ...];
```

- **Cách 2:** (Gọi mảng theo Object)

```
var arrayName = new Array(item1, item2, ...);
```

2. Chuyển đổi dữ liệu từ Arrays sang String: (toString())

```
var arrayName = [Item1, Item2, Item3, Item3];
document.getElementById("demo").innerHTML = arrayName.toString();
→ Item1, Item2, Item3, Item4
```

3. Một số phương thức quan trọng:

- **Xử lý phần tử trong mảng:**
  - **array.join("phép nối")** → Đưa các phần tử trong mảng về dạng chuỗi
  - **array.pop()** → Xóa phần tử cuối cùng của mảng
  - **array.push(newItem)** → Thêm phần tử newItem vào cuối mảng
  - **array.shift()** → Xóa phần tử đầu tiên của mảng
  - **array.unshift(newItem)** → Thêm phần tử newItem vào đầu mảng
  - **delete array[index]** → Xóa phần tử ở vị trí index trong mảng
  - **array1.concat(array2)** → Ghép hai mảng thành một mảng chung

- **Thêm hoặc bớt phần tử vào mảng ở vị trí bất kỳ:**
  - ***array.splice(index, clearIndex, addItem1, addItem2, ...);***
    - **index:** chỉ số của phần tử đó trong mảng
    - **clearIndex:** xóa bao nhiêu phần tử từ vị trí index
    - **addItem:** phần tử muốn thêm
  - Thêm phần tử ở vị trí index → ***array.splice(index, 0, addItem)***
  - Xóa k phần tử từ vị trí index → ***array.splice(index, k)***

- **Sắp xếp mảng dữ liệu:**
  - Sắp xếp tăng dần các phần tử trong mảng: ***array.sort();***
  - Đảo ngược các phần tử trong mảng: ***array.reverse();***

BÀI 9: HÀM FUNCTION

1. Khởi tạo hàm Function():

```
function myFunction(paramter1, parameter2, ...) {
    // Thực thi các chức năng trong hàm
    return result;
}
```

- Ta nên đặt tên function() là các động từ: (hàm bắt đầu với các từ)
    - **getFunction()** → trả về một giá trị
    - **calcFunction()** → tính toán cái gì đó
    - **createFunction()** → tạo ra cái gì đó
    - **checkFunction()** → kiểm tra cái gì đó và trả về Boolean

2. Function lồng Function (Nested Functions):

```
function myFunction1(paramter1, parameter2, ...) {
        var nameValue = 0;
        function plus() {
            // Những thay đổi về nameValue;
        }
        plus();
        return nameValue;    → nameValue này sẽ mang giá trị mới
}
```

3. Gọi Function từ JavaScript tới HTML:

```
<thẻ onclick = "functionName()"> Content </thẻ>
<script>
    // Hoạt động của các Function JavaScript
    function nameFunction1() {
        // Những hành động của nameFunction1
    }

    function nameFunction2() {
        // Những hành động của nameFunction2
    }

    ...
</script>
```

BÀI 10: HƯỚNG ĐỐI TƯỢNG JS

1. Đối tượng trong JavaScript:

- Lưu trữ các thuộc tính (cặp key – value) với key là chuỗi hoặc ký hiệu, value là giá trị bất kỳ ứng với key
- Cách truy cập một thuộc tính:
  - C1 (ký hiệu dấu chấm) → ***obj.property = propertyValue;***
  - C2 (ký hiệu ngoặc vuông) → ***obj[propertyName]***
- Toán tử đối tượng Object:
  - Xóa thuộc tính → ***delete obj.prop***
  - Kiểm tra thuộc tính có từ khóa nào tồn tại không → ***"key" in obj***
  - Lặp đối tượng bằng for → ***for (let key in obj) { }***

2. Phương thức của đối tượng và "this" trong JavaScript:

- Các hàm được lưu trữ trong các thuộc tính đối tượng được gọi là các phương thức và chúng thực hiện hành động của đối tượng → ***object.doSomething();***
- Các phương thức có thể tham chiếu đối tượng như ***this***

3. Constructor và Operator "new" trong JavaScript:

- Giống trong Java, các hàm Constructor được khởi tạo mặc định trong lập trình hướng đối tượng
- Hàm Constructor chỉ nên được gọi bằng cách sử dụng ***new*** và nó sẽ ngụ ý tạo ra sản phẩm ***this*** và trả về cho đối tượng
- Ta có thể sử dụng các hàm Constructor để tạo nhiều đối tượng tương tự:
  - **Encapsulation**: Tính đóng gói (Khả năng lưu giữ thông tin liên quan, là dữ liệu hoặc các phương thức, cùng với nhau trong một đối tượng)
  - **Abstraction**: Tính trừu tượng (Khả năng lưu trữ đối tượng này bên trong đối tượng khác)
  - **Inheritance**: Tính kế thừa (Khả năng của một lớp dựa trên lớp khác đối với thuộc tính và phương thức của nó)
  - **Polymorphism**: Tính đa hình (Khả năng viết hàm hoặc phương thức làm việc đa dạng nhiều cách khác nhau)

4. Khởi tạo đối tượng bằng toán tử New:

- Toán tử new được sử dụng để tạo đối tượng với Object(), Array(), Date():
  - ***var object1 = new Object();*** → Xây dựng đối tượng chung (biến object1 chứa tham chiếu tới đối tượng mới)

Tạo Object	Mô tả

Object Dictionary	var nameDict = {
	key1: 'value1',
	key2: 'value2',
	...
	};

Object Create (body)	var object1 = new Object(); object1.elementA = <value1>; object2.elementB = <value2>;

Object Constructor (head)	function object(element1, element2) {
	this.element1 = element1;
	this.element2 = element2;
	}

Định nghĩa phương thức cho đối tượng (head)	+ Cách 1:
	function addValue(value) {
	this.element = value;
	}
	
	+ Cách 2:
	function addValue(value) {
	with (this) {
	element = value;
	}
	}
	
	+ Constructor
	function object(element1, element2) {
	this.element1 = element1;
	this.element2 = element2;
	this.element3 = addValue;
	}

- var object2 = new Date("Moth Day, Year");
  - Mặc định → ***var objectName = new Date();*** => Trả về thời gian hiện tại
  - Thời gian → ***var objectName = new Date(miliseconds);***
  - Chuỗi → ***var objectName = new Date(dateString);*** => với chuỗi được ***Date.parse()*** tương ứng
  - Tổng hợp → ***var objectName = new Date(year, moth, date[hour, minute, second, milisecond]);***
    => Có các hàm getter và setter tương ứng khi khởi tạo Constructor Date

BÀI 11: CÁC HÀM TOÁN HỌC

1. Khai báo đối tượng Math:

- Cú pháp: var ***nameMath = Math.<thuộc tính>;***

2. Các thuộc tính của Math:

Thuộc tính	Mô tả
Math.E	Hằng số e = 2,718...
Math.LN2	Ln(2) = 0.693...
Math.LN10	Ln(10) = 2.302...
Math.LOG2E	Logarit cơ số 2 của e = 1.442...
Math.LOG10E	Logarit cơ số 10 của e = 0.434...
Math.PI	Số PI = 3.14159...
Math.SQRT1_2	Căn bậc hai của ½
Math.SQRT2	Căn bậc hai của 2

3. Các phương thức của Math:

Phương thức	Mô tả
Math.abs(value)	Giá trị tuyệt đối của value
Math.acos(value)	Giá trị arccos(value) theo radians
Math.asin(value)	Giá trị arcsin(value) theo radians
Math.atan(value)	Giá trị arctan(value) theo radians
Math.cos(value)	Giá trị cos(value) theo radians
Math.sin(value)	Giá trị sin(value) theo radians
Math.tan(value)	Giá trị tan(value) theo radians
Math.cell(value)	Làm tròn số lên trên
Math.floor(value)	Làm tròn số xuống dưới
Math.round(value)	Làm tròn số gần nhất
Math.exp(value)	Giá trị e^value
Math.log(value)	Logarit cơ số e của value
Math.max(val1, val2, ...)	Trả về số lớn nhất
Math.min(val1, val2, ...)	Trả về số nhỏ nhất
Math.pow(cơ số, số mũ)	Trả về giá trị (cơ số)^(số mũ)
Math.random()	Trả về số ngẫu nhiên giữa số 0 và 1
Math.sqrt(number)	Căn bậc hai của number
toSource()	Trả về chuỗi “Math”

BÀI 12: THƯ VIỆN REACT JS

1. **Kiểm tra các phiên bản NodeJS trên Terminal:**
   - node -v
   - npm -v
   - npx -v

2. **Tạo project:**
   - B1: Chọn 1 thư mục Folder bất kì
   - B2: Tải gói lệnh môi trường → `npm install -g create-react-app`
   - B3: Tạo project bằng lệnh:
     - `create-react-app my-app`
     - npx create-react-app <nameMyApp>
   - B4: Mở project vừa tạo và chạy lệnh → `npm start (ở mục demo)`

3. **Tạo các Components:**
   - Tạo folder components trong thư mục src
   - Các đối tượng lấy chung đều nằm trong folder components (file.js hoặc file.css)
     - Cấu trúc componentFile.js:

```
import './componentFile.css'
function componentFile(props) {
        return <div className = "name" style = {{backgroundColor:props.color}}>
        # Các thẻ khác nếu có
       </div>
}
export {componentFile};
```

     - Cấu trúc componentFile.css:

```
.name {
     # Các thuộc tính của CSS như height, width, border, padding, margin, ...
    # Các thuộc tính căn chữ như text-align (căn vị trí chữ), border-radius (đường viền cong), ...
    # Các thuộc tính phông chữ font-size, font-weight, ...
```

```
}
```

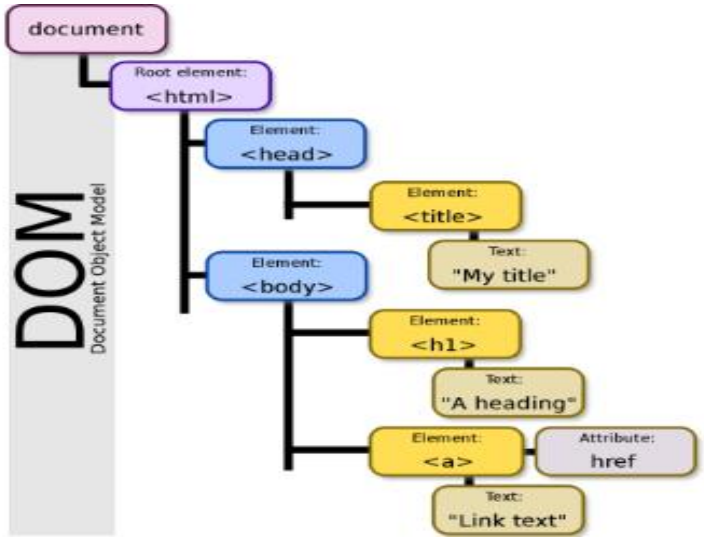
- Cấu trúc file main App.js:

```
// Import thư viện
import './App.css';
import { componentFile } from './components/ componentFile ';

// Hàm main
function App() {
    return (
        <div className = “App”>
            // Đối tượng khởi tạo
            <componentFile
                name = ...;
                color = ...;
            />
    )
}
```

BÀI 13: DOM TRONG JAVASCRIPT

1. **Khái niệm DOM:**

- Document Object Model (Mô hình các đối tượng tài liệu) → Truy xuất và thao tác trên các tài liệu có cấu trúc dạng HTML và XML bằng các ngôn ngữ lập trình Javascript, PHP, ...
- Cấu trúc DOM:
    - Mọi thành phần đều được xem là một Node (nút) và biểu diễn trên cây cấu trúc gọi là DOM Tree
    - Các phần tử khác nhau sẽ được phân loại node khác nhau:
        - Node gốc → document node (là HTML biểu diễn bởi thẻ <html>)
        - Node phần tử → element node (biểu diễn cho một phần tử HTML)
        - Node văn bản → text node (mỗi đoạn ký tự bên trong HTML: <title>, <h1>, <p>)
        - Node thuộc tính → attribute node
        - Node chú thích → comment node
- Sơ đồ DOM:



2. **DOM trong Javascript:**

- DOM document → lưu trữ toàn bộ các thành phần trong documents của website

- DOM element → truy xuất tới thẻ HTML nào đó thông qua các thuộc tính như tên class, id, name của thẻ HTML
- DOM HTML → thay đổi giá trị nội dung và giá trị thuộc tính của các thẻ HTML
- DOM CSS → thay đổi các định dạng CSS của thẻ HTML
- DOM Event → gán các sự kiện như onclick(), onload() vào các thẻ HTML
- DOM Listener → lắng nghe các sự kiện tác động lên thẻ HTML
- DOM Navigation → quản lý, thao tác với các thẻ HTML, thể hiện mối quan hệ cha – con của thẻ HTML
- DOM Node, Nodelist → thao tác với HTML thông qua đối tượng Object

3. <u>Các thuộc tính DOM:</u>

- id → Định danh → duy nhất cho mỗi phần tử nên được dùng để truy xuất DOM trực tiếp và nhanh chóng
- className → Tên lớp → dùng để truy xuất trực tiếp như id, một class có thể dùng cho nhiều phần tử
- tagName → tên thẻ HTML
- innerHTML → trả về mã HTML bên trong phần tử hiện tại (chuỗi ký tự chứa các phần tử bên trong: node phần tử, node văn bản)
- outerHTML → trả về mã HTML của phần tử hiện tại
  => outerHTML = tagName + innerHTML
- textContent → trả về chuỗi ký tự chứa nội dung của tất cả node văn bản bên trong phần tử hiện tại
- attributes → tập các thuộc tính như id, name, class, href, title, ...
- style → tập các định dạng của phần tử hiện tại
- value → lấy giá trị của thành phần được chọn thành một biến

4. <u>Các phương thức DOM:</u>

- getElementById(id) → tham chiếu đến 1 node duy nhất có thuộc tính id giống với id cần tìm
- getElementsByTagName(tagname) → Tham chiếu đến tất cả các node có thuộc tính tagName giống với tên thẻ cần tìm

- Nếu muốn truy xuất đến toàn bộ thẻ trong tài liệu HTML thì hãy sử dụng → document.getElementsByTagName('*')
- getElementsByName(name) → Tham chiếu đến tất cả các node có thuộc tính name cần tìm
- getAttribute(attributeName) → Lấy giá trị của thuộc tính
- setAttribute(attributeName, value) → Sửa giá trị của thuộc tính
- appendChild(node) → Thêm 1 node con vào node hiện tại
- removeChild(node) → Xóa 1 node con khỏi node hiện tại
- focus() → đưa trở lại giá trị ban đầu

5. **Truy xuất DOM:**
    - document.getElementById()
    - document.getElementsByTagName()
    - document.getElementsByClass()
    - document.getElementsByName()

BÀI 14: THƯ VIỆN CSS BOOTSTRAP

1. **Khái niệm BOOTSTRAP:**
   - Là Framework HTML, CSS, JavaScript cho phép thiết kế, phát triển trang web hỗ trợ responsive để tạo giao diện
   - Ưu điểm:
     - Chuẩn hiển thị trên mọi màn hình, trình duyệt
     - Tự động điều chỉnh kích thước theo từng ứng dụng
     - Tốc độ load nhanh
     - Hỗ trợ giao diện sẵn, kiểu box, form đẹp
     - Miễn phí và mã nguồn mở hướng vào phát triển website đầu tiên đáp ứng thiết bị di động
   - Set up Bootstrap:
     - CSS → <link href = "folder/linkBootstrap.css">
     - JavaScript → <script src = "folder/linkBootstrap.js"

2. **Bootstrap COLOR:**
   - Có một số class theo ngữ cảnh, được sử dụng để ngầm biểu đạt một ý nghĩa qua màu sắc
     - Các class cho màu sắc của chữ gồm có: .text-muted, .text-primary, .text-success, .text-info, .text-warning, .text-danger, .text-secondary, .text-white , .text-dark , .text-body (màu body mặc định thường là đen) và .text-light
     - Class cho màu nền gồm có: .bg-primary, .bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark và .bg-light
     - Lưu ý rằng, màu nền sẽ không thiết lập màu chữ → vì vậy, trong một số trường hợp, bạn sẽ cần sử dụng class màu nền cùng với class .text-*

Màu chữ

```
<div class="container mt-3">
  <h2>Màu sắc theo ngữ cảnh</h2>
  <p>Sử dụng các class theo ngữ cảnh để cung cấp những ý nghĩa thông qua màu
  <p class="text-muted">Văn bản bị muted.</p>
  <p class="text-primary">Văn bản quan trọng.</p>
  <p class="text-success">Văn bản chỉ ra sự thành công.</p>
  <p class="text-info">Văn bản đại diện cho một số thông tin.</p>
  <p class="text-warning">Văn bản này là cảnh báo.</p>
  <p class="text-danger">Văn bản thông báo sự nguy hiểm.</p>
  <p class="text-secondary">Văn bản phụ.</p>
  <p class="text-dark">Văn bản này có màu xám đen.</p>
  <p class="text-body">Màu body mặc định (thường là màu đen).</p>
  <p class="text-light">Văn bản màu xám nhạt (trên nền trắng).</p>
  <p class="text-white">Văn bản màu trắng (trên nền trắng).</p>
</div>
```

Màu nền

```
<div class="container mt-3">
  <h2>Màu nền theo ngữ cảnh</h2>
  <p>Sử dụng các class màu nền theo ngữ cảnh.</p>
  <p class="bg-primary text-white">Văn bản quan trọng.</p>
  <p class="bg-success text-white">Văn bản chỉ ra sự thành công.
  <p class="bg-info text-white">Văn bản chứa thông tin.</p>
  <p class="bg-warning text-white">Văn bản này là cảnh báo.</p>
  <p class="bg-danger text-white">Văn bản thông báo sự nguy hiểm
  <p class="bg-secondary text-white">Màu nền phụ.</p>
  <p class="bg-dark text-white">Màu nền xám đen.</p>
  <p class="bg-light text-dark">Màu nền xám trắng.</p>
</div>
```

3. **Bootstrap BREAK POINT và CONTAINER:**

- Bảng Break Point: tùy theo kích cỡ thì giao diện sẽ theo hình hiển thị trên màn hình được chỉ định (Responsive)

Breakpoint	Class infix	Dimensions
Extra small	*None*	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large	xl	≥1200px
Extra extra large	xxl	≥1400px

- Class Container: Reponsive lại trang web tùy vào kích cỡ Bootstrap Break Point (căn chỉnh theo độ rộng)

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px	XX-Large ≥1400px
.container	100%	540px	720px	960px	1140px	1320px
.container-sm	100%	540px	720px	960px	1140px	1320px
.container-md	100%	100%	720px	960px	1140px	1320px
.container-lg	100%	100%	100%	960px	1140px	1320px
.container-xl	100%	100%	100%	100%	1140px	1320px
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

- Là một phần tử có thể chứa nhiều phần tử khác (, , ...) → lớp .container hay .container-fluid sẽ được sử dụng cho các phần tử này
  → Viết trong thẻ và gọi class = “container” trong thẻ đó (chú ý nhớ gán link sử dụng Bootstrap)
  → Ta có class “col” và class “row” là có sẵn cho cột và hàng

Start aligned text on all viewport sizes.

Center aligned text on all viewport sizes.

End aligned text on all viewport sizes.

Start aligned text on viewports sized SM (small) or wider.

Start aligned text on viewports sized MD (medium) or wider.

Start aligned text on viewports sized LG (large) or wider.

Start aligned text on viewports sized XL (extra-large) or wider.

HTML

```
<p class="text-start">Start aligned text on all viewport sizes.</p>
<p class="text-center">Center aligned text on all viewport sizes.</p>
<p class="text-end">End aligned text on all viewport sizes.</p>

<p class="text-sm-start">Start aligned text on viewports sized SM (small) or wider.</p>
<p class="text-md-start">Start aligned text on viewports sized MD (medium) or wider.</p>
<p class="text-lg-start">Start aligned text on viewports sized LG (large) or wider.</p>
<p class="text-xl-start">Start aligned text on viewports sized XL (extra-large) or wider.</p>
```

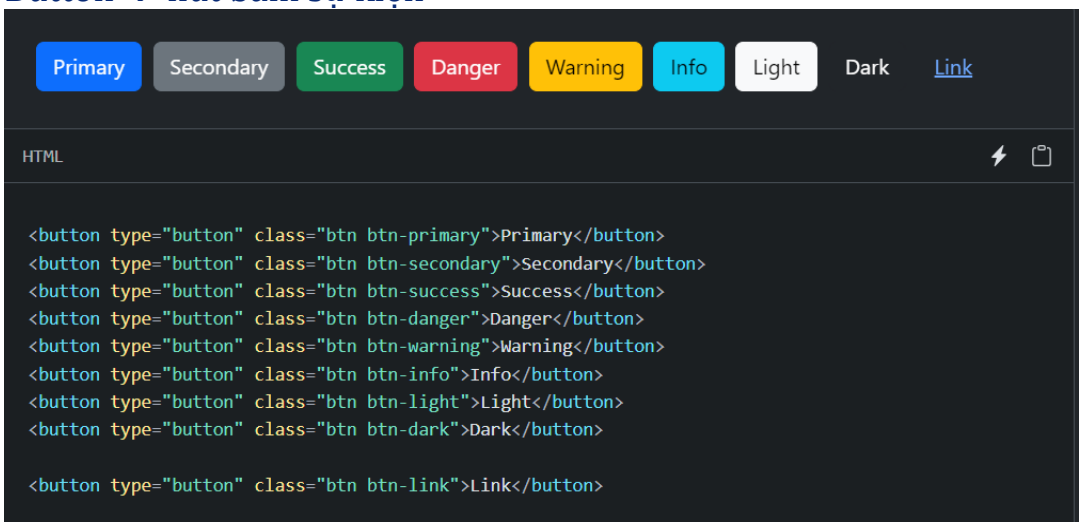
4. **Bootstrap GRID SYSTEM:**
   - Là một trong những định nghĩa, thành phần quan trọng của Bootstrap → giúp bố trí các thành phần trên giao diện
   - Website có thể tương thích với nhiều giao diện trên thiết bị khác nhau
   - Được xây dựng flexbox tạo tới 12 cột trên một trang (riêng biệt hoặc gộp)
   - Có thể responsive các cột tự sắp xếp lại phù hợp với kích cỡ màn hình

1 cột	1 cột	1 cột	1 cột	1 cột	1 cột	1 cột	1 cột	1 cột	1 cột	1 cột	1 cột
gộp 4 cột				gộp 4 cột				gộp 4 cột			
gộp 4 cột				gộp 8 cột							
gộp 6 cột						gộp 6 cột					
gộp 12 cột											

5. **Các loại Bootstrap khác:**
   - Link tham khảo:
     https://getbootstrap.com/docs/5.3/components/buttons/
     https://getbootstrap.com/docs/4.0/utilities/embed/

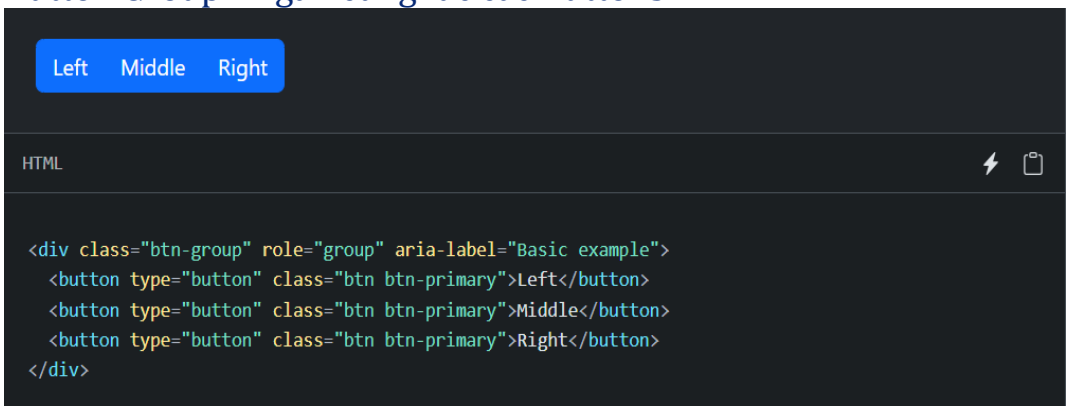
   - Bootstrap Component Basic:

- Button → nút bấm sự kiện



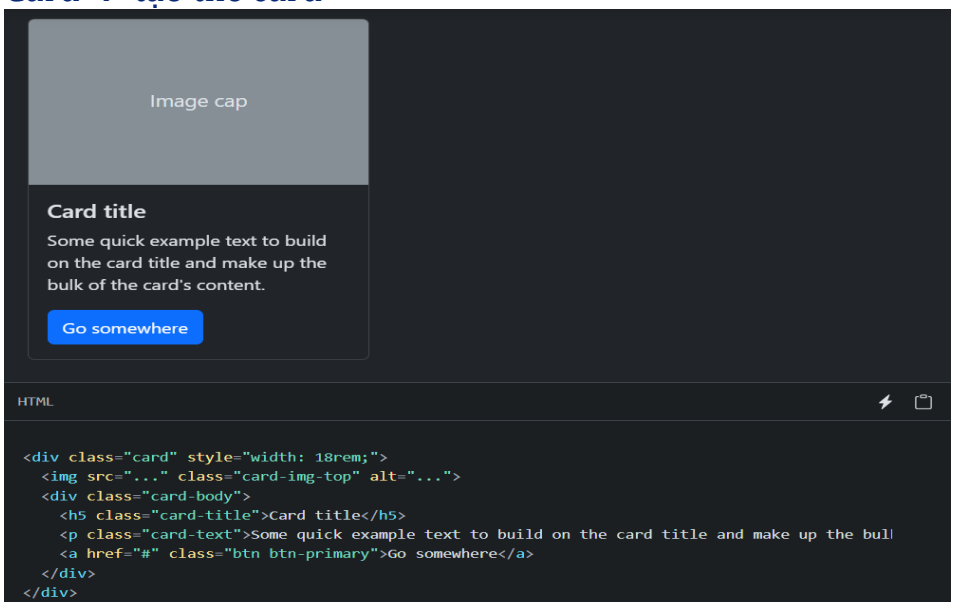

```html
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>
```

- Button Group → gán cùng lúc các Buttons




```html
<div class="btn-group" role="group" aria-label="Basic example">
  <button type="button" class="btn btn-primary">Left</button>
  <button type="button" class="btn btn-primary">Middle</button>
  <button type="button" class="btn btn-primary">Right</button>
</div>
```

- Card → tạo thẻ card




```html
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bul
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

- Một số thẻ khác:
  - Carousel → tạo slide cho web
  - Navbar → tạo menu
  - Form → gửi thông tin về máy chủ
  - List
  - Select
  - Check & Radio

- Bootstrap Table:
  - Cần áp dụng sử dụng lớp .table cho các thẻ (tag) <table> và kèm với một vài lớp bổ sung để có thể tạo ra được bảng theo ý muốn → sử dụng rộng rãi trong website
  - Với lớp .table cho <Table> thì có thể tạo được một bảng cơ bản nhất theo phong cách Bootstrap

- Bootstrap NavBar:
  - Là một phần giao diện của giao diện người dùng
  - Giúp người sử dụng có thể thấy và di chuyển đến các trang khác trong website

- Bootstrap Modal:
  - Là hộp thoại (Dialog) hay cửa sổ bật lên (popup), khi đó sẽ hiển thị phía trên tất cả các nội dung khác trên một page hiện tại
  - Mục đích của Bootstrap Modal là thông báo cho người dùng thông tin xác nhận

- Bootstrap Form:
  - Là thành phần các form được tạo ra trong giao diện website
  - Từ form có thể điền được các thông tin quan trọng
  - Form có thể mô tả được các thông tin mà muốn người dùng nắm bắt được

- Bootstrap Input Group:
  - Là phương thức mở rộng một Input Control bằng phương pháp thêm vào bên cạnh nó các phần tử
  - Các phần tử được them: Text, Button hay Button Group
  - Các phần tử được thêm vào bênh cạnh Input Control sẽ được gọi là các Addons

- Bootstrap Pagination: (phân trang phía cuối)
    - Với 1 page có nhiều dữ liệu, thông tin, hay nó quá dài và lớn để có thể hiển thị trên 1 page, cần phải chia nó ra làm nhiều phần, mỗi phần đó sẽ là 1 page và nó sẽ hiển thị 1 vài dữ liệu, thông tin
    - Khi đó bạn sẽ cần đến 1 link để cho người dùng có thể click đến những trang tiếp đó.
    - Những link đó chính là Bootstrap Pagination

- Bootstrap Badge: (huy hiệu thông báo)
    - Là thành phần giao diện nhỏ và được sử dụng trang trí cho 1 thành phần nào đó
    - Sử dụng lớp .badge áp dụng cho 1 thẻ tag để có thể tạo ra 1 Badge
    - Một vài lớp bổ sung như .badge-primary, .badge-secondary, .badge-danger, ...

- Bootstrap Dropdown:
    - Là một component và nó bao gồm 1 button, menu
    - Khi người dùng nhấn vào Button thì menu sẽ xuất hiện
    - Cho phép người dùng có thể lựa chọn giá trị từ 1 danh sách

- Bootstrap Alert:
    - Là một thành phần giao diện đã được xây dựng sẵn trong Bootstrap
    - Là một vùng không gian sẽ hiển thị 1 thông điệp như thông tin, cảnh báo lỗi