Lập trình Front-end Web 1

| Front-end Web Development 1|

Nguyễn Huy Hoàng – Bùi Thị Phương Thảo

[02 . 2019]

Dùng kèm Lập trình Web Front-end 1, Khoa Công nghệ thông tin, Cao đẳng Công nghệ Thủ Đức

FACULTY OF
INFORMATION TECHNOLOGY
THU DUC COLLEGE OF TECHNOLOGY

Dẫn nhập

- Ngôn ngữ lập trình ở front-end web gồm những ngôn ngữ nào?

Bài 4.

TỔNG QUAN VỀ JAVASCRIPT

4.1

Giới thiệu JavaScript

Giới thiệu JavaScript

- JavaScript != Java
- Được tạo ra bởi Brendan Eich vào 1995 (Mocha -> LiveScript -> JavaScript)
- Trở thành tiêu chuẩn ECMA vào 1997.
- Là scripting language (ngôn ngữ kịch bản: ngôn ngữ thực hiện các hành động cho các đối tượng)
- Phiên bản JavaScript theo đặc tả của ECMAScript. Bản mới nhất hiện nay là bản thứ 9 (ECMAScript 2018)

Giới thiệu JavaScript

- Là ngôn ngữ ở cả client & server.

⇒Học phần Front-end 1: chỉ nói về JavaScript ở client.

JavaScript làm được gì?

- Thay đổi HTML.
- Thay đổi CSS.
- Bắt các sự kiện của người dùng (click, key, hover, drag & drop, v.v...)
- Tạo ra các chuyển động, hiệu ứng, game, v.v...

4.2

Cú pháp cơ bản của JavaScript

Viết JavaScript

- Internal: các lệnh JS đặt trong cặp thẻ **<script></script>**. Cặp thẻ này nên đặt ở trước </body>
- External: các lệnh JS đặt trong file *.js. File này sau đó được gọi thông qua thẻ **<script src="đường dẫn file *.js"></script>**

Các quy ước chung

- Tên biến và tên hàm viết theo kiểm **camelCase**
- Có một khoảng trắng trước & sau toán tử = + - * / và sau dấu ,
- Lùi đầu dòng: 4 khoảng trắng.
- Kết thúc câu lệnh bằng ;
- Một dòng lệnh không nên vượt quá 80 ký tự. Nếu dài quá, nên ngắt sau dấu , hoặc sau toán tử.

Chú thích

//: chú thích một dòng

/* */: chú thích nhiều dòng. /* đặt ở dòng đầu tiên, */ đặt ở dòng cuối cùng

Câu lệnh JavaScript đầu tiên

```
alert('Hello World');
```

Hãy thử và cảm nhận ^^

Biến

```
let tenBien;
let tenBien = giátrịbanđầu;
```

- Tên biến: camelCase, a-zA-Z0-9_$ phải bắt đầu bằng ký tự chữ hoặc _
- Khai báo nhiều biến: các biến cách nhau bằng dấu ,
- Một biến có thể là bất kỳ kiểu dữ liệu nào, tùy vào kiểu dữ liệu của giá trị biến đó mang

Tầm ảnh hưởng của biến

- **Global**: được khai báo ngoài các hàm. Phạm vi hoạt động: từ vị trí khai báo trở về sau trong trang web. Biến bị hủy khi đóng tab hoặc đóng trình duyệt.
  Biến được sử dụng trong hàm nhưng chưa khai báo sẽ được tự động trở thành biến Global.
- **Local**: được khai báo trong hàm. Phạm vi hoạt động của biến là từ vị trí khai báo đến kết thúc hàm. Biến bị hủy khi hàm kết thúc.

Mảng

```
let tenMang = [item1, item2, ...];
```

- Một mảng có n phần tử được đánh chỉ số từ 0 đến n-1.
- Để lấy ra số phần tử của mảng:

```
tenMang.length
```

Mảng

Vòng lặp forEach()

```
tenMang.forEach(function(item, index, array) {
    // ... do something with item
});
```

Đối tượng (Object)

```
let tenDoiTuong = {
  propertyName: value,
  propertyName: value,
  methodName: function() {
        //Hành động của đối tượng
  }
};
```

- Đối tượng gồm properties và methods

Hằng

```
const tenHang = giátrị;
```

4.3

Hàm trong JavaScript

Function

```
function functionName (parameter, parameter = defaultvalue) {

}
```

- Tên function: camelCase.

Function Expression

```
let functionName = function(parameter, parameter = defaultvalue)
{

}
```

Arrow function

```
let functionName = (parameter, parameter = defaultvalue) => {

}
```

4.4

Browser events

Mouse events

- **click** – when the mouse clicks on an element (touchscreen devices generate it on a tap).
- **contextmenu** – when the mouse right-clicks on an element.
- **mouseover / mouseout** – when the mouse cursor comes over / leaves an element.
- **mousedown / mouseup** – when the mouse button is pressed / released over an element.
- **mousemove** – when the mouse is moved.

Keyboard events

- **keydown and keyup** – when a keyboard key is pressed and released.

Form element events

- **submit** – when the visitor submits a <form>.
- **focus** – when the visitor focuses on an element, e.g. on an <input>.

Document events

- **DOMContentLoaded** – when the HTML is loaded and processed, DOM is fully built.

CSS events

- **transitionend** – when a CSS-animation finishes.

addEventListener

element.**addEventListener**(event, handler, [options]);

- **event**: Event name, e.g. "click”
- **handler:** The handler function
- **options**: An additional optional object with properties: once, capture, passive

Scheduling: setTimeout and setInterval

setTimeout allows us to run a function once after the interval of time.

- let timerId = **setTimeout**(func|code, [delay], [arg1], [arg2], ...);
- **clearTimeout**(timerId);

setInterval allows us to run a function repeatedly, starting after the interval of time, then repeating continuously at that interval.

- let timerId = **setInterval**(func|code, [delay], [arg1], [arg2], ...);
- **clearInterval**(timerId);

Scheduling: setTimeout and setInterval

```
function sayHi() {
alert('Hello');
}
setTimeout(sayHi, 1000);
```

```
// wrong!
setTimeout(sayHi(), 1000);
```

```
function sayHi(phrase, who) {
alert( phrase + ', ' + who );
}
setTimeout(sayHi, 1000, "Hello", "John"); // Hello, John
```

```
setTimeout("alert('Hello')", 1000); //works but not recommended
```

```
setTimeout(() => alert('Hello'), 1000);
```

Nested setTimeout

There are two ways of running something regularly: One is setInterval. The other one is a nested setTimeout.

```
/** instead of:
let timerId = setInterval(() => alert('tick'), 2000);
*/

let timerId = setTimeout(function tick() {
       alert('tick’);
       timerId = setTimeout(tick, 2000); // (*)
}, 2000);
```

4.5

DOM

getElement*, querySelector*

There are 6 main methods to search for nodes in DOM:

Method	Searches by...	Can call on an element?	Live?
`querySelector`	CSS-selector	✔	-
`querySelectorAll`	CSS-selector	✔	-
`getElementById`	`id`	-	-
`getElementsByName`	`name`	-	✔
`getElementsByTagName`	tag or `'*'`	✔	✔
`getElementsByClassName`	class	✔	✔

Xử lý HTML

Các hàm lấy nội dung HTML

- **innerHTML**: Lấy/Gán nội dung bất kỳ cho thẻ HTML được chọn. (nội dung có thể bao gồm chuỗi HTML).
- **innerText**: Lấy/Gán nội dung text cho thẻ được chọn (trừ thẻ <script>, <style> và các thẻ bị css ẩn).
- **textContent**: Lấy/Gán nội dung text cho thẻ được chọn.
- **value**: Lấy giá trị value của các thành phần trong form.

Attribute

- **elem.hasAttribute(name)**: to check for existence.
- **elem.getAttribute(name)**: to get the value.
- **elem.setAttribute(name, value)**: to set the value.
- **elem.removeAttribute(name)**: to remove the attribute.
- **elem.attributes** is a collection of all attributes.

Styles

To change the styles:

- The **style** property is an object with camelCased styles. Reading and writing to it has the same meaning as modifying individual properties in the "style" attribute. To see how to apply important and other rare stuff – there's a list of methods at MDN.
- The **style.cssText** property corresponds to the whole "style" attribute, the full string of styles.

To read the resolved styles (with respect to all classes, after all CSS is applied and final values are calculated):

- The **getComputedStyle(elem, [pseudo])** returns the style-like object with them. Read-only.

Classes

To manage classes, there are two DOM properties:

- **className** – the string value, good to manage the whole set of classes.
- **classList** – the object with methods **add/remove/toggle/contains**, good for individual classes.

Methods of classList:

- **elem.classList.add/remove("class")** – adds/removes the class.
- **elem.classList.toggle("class")** – adds the class if it doesn't exist, otherwise removes it.
- **elem.classList.contains("class")** – checks for the given class, returns true/false.

Attribute vs Properties

	Properties	Attributes
Type	Any value, standard properties have types described in the spec	A string
Name	Name is case-sensitive	Name is not case-sensitive

```
<a id="a" href="#hello">link</a>
<script>
// attribute
alert(a.getAttribute('href')); // #hello
// property
alert(a.href ); // full URL in the form http://site.com/page#hello
</script>
```

Create new nodes

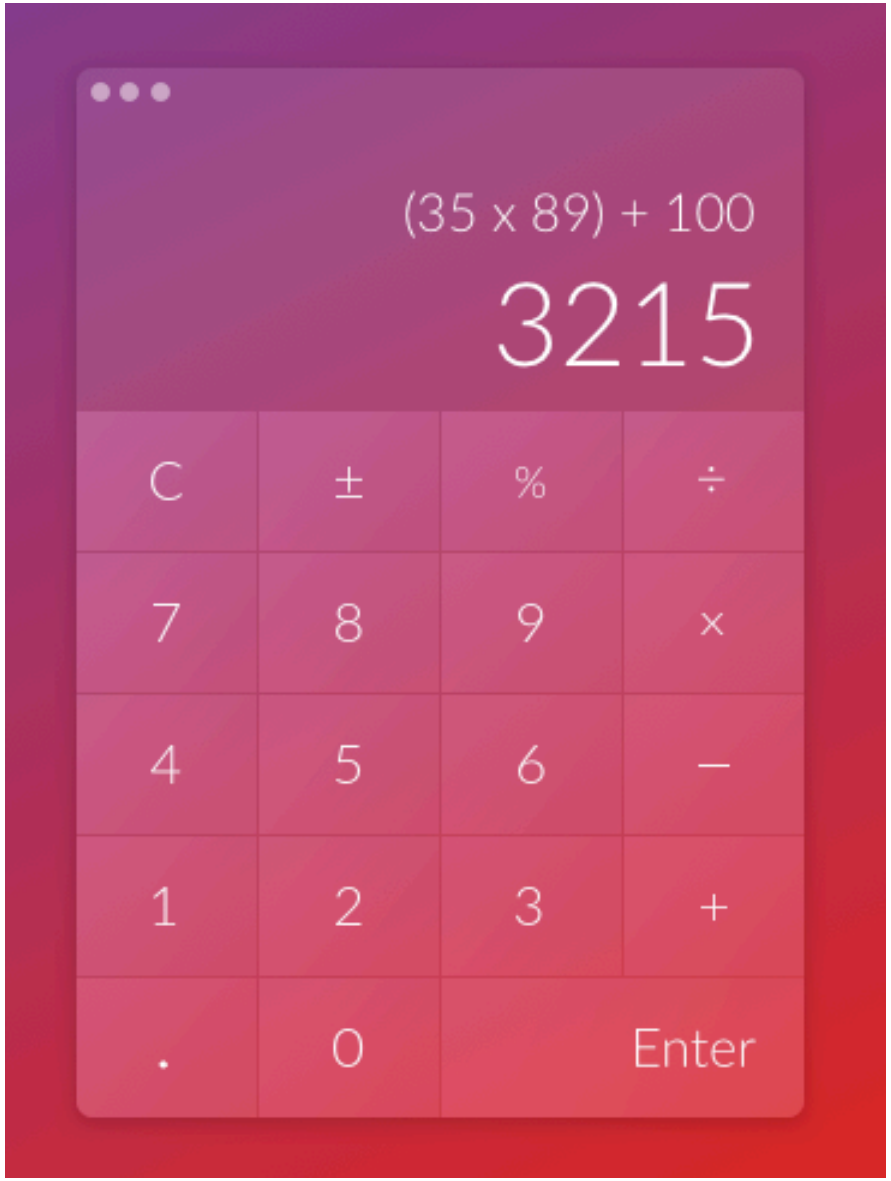
- **document.createElement(tag)** – creates an element with the given tag,
- **document.createTextNode(value)** – creates a text node (rarely used),
- **elem.cloneNode(deep)** – clones the element, if deep==true then with all descendants.

Insertion and removal

- **node.append(...nodes or strings)** – insert into node, at the end,
- **node.prepend(...nodes or strings)** – insert into node, at the beginning,
- **node.before(...nodes or strings)** –- insert right before node,
- **node.after(...nodes or strings)** –- insert right after node,
- **node.replaceWith(...nodes or strings)** –- replace node.
- **node.remove()** –- remove the node.

Bài tập

1. Máy tính điện tử



Bài tập

2. Đồng hồ số

Bài tập

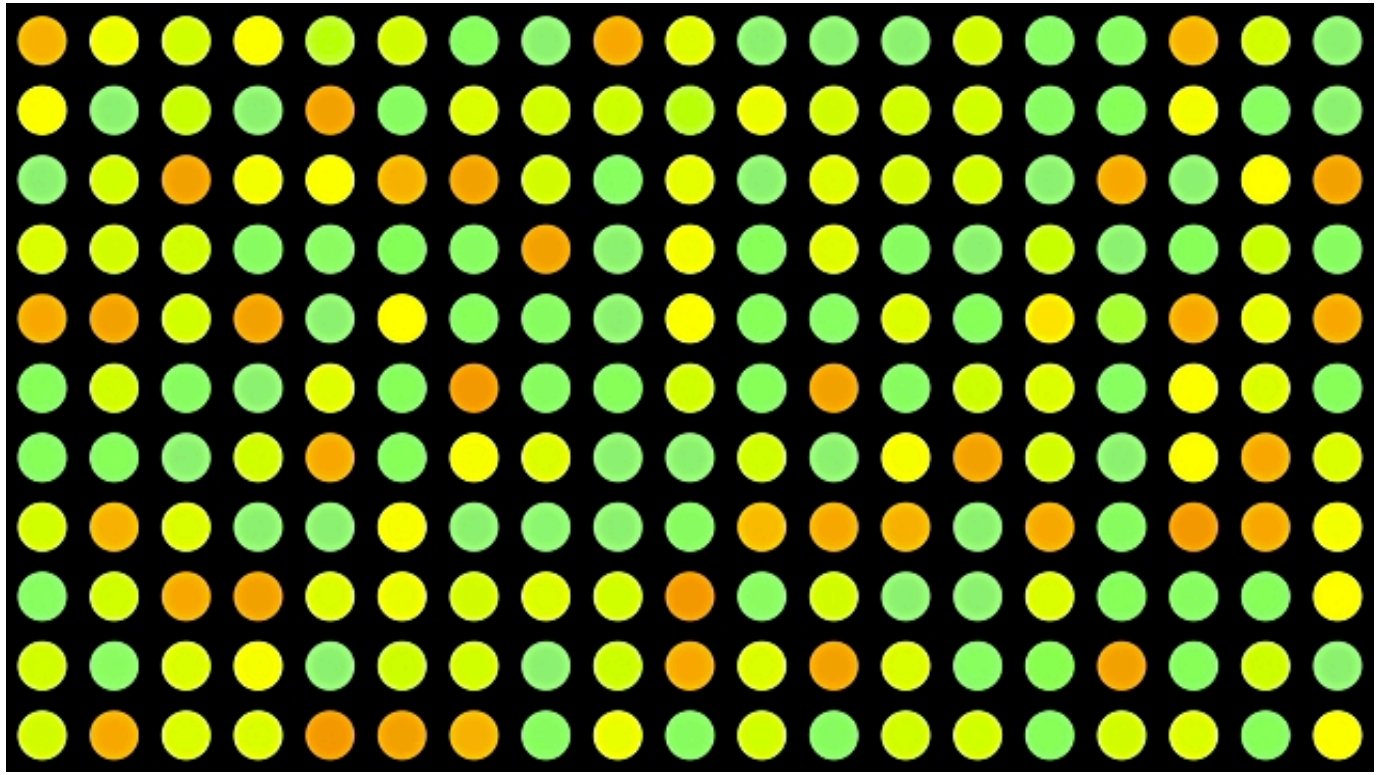
3. Chong chóng:
3.1. Tự quay
3.2. Có nút Start / Stop

Bài tập

4. Đồng hồ kim

Bài tập

5. Đèn chớp tắt đổi màu



Bài tập

6. Thực hiện trò chơi sau:

Cho trước 10 hình tròn (30 x 30px) với 5 màu sau (mỗi màu 2 hình):

- Đỏ: 10 điểm.
- Xanh dương: 5 điểm.
- Xanh lá: 2 điểm.
- Cam: 1 điểm.
- Đen: 0 điểm.

Khi click nút bắt đầu chơi, cứ mỗi 0.5 giây, vị trí của các hình tròn sẽ được xáo trộn ngẫu nhiên trong khoảng:

- Chiều ngang: 0 - 1200px.
- Chiều dọc: 0 - 800px.

Người chơi sẽ click lên các hình tròn, số điểm của hình tròn tương ứng sẽ được cộng dồn vào tổng điểm của người chơi và hiển thị lên màn hình. Người chơi có 10 giây để chơi. Khi hết 10 giây, một thông báo Hết giờ hiện ra và cho biết tổng điểm cuối cùng của người chơi.

Bài tập

7. Game tìm số. Khi click Start, game tự phát sinh ra 100 số ở vị trí bất kỳ trên màn hình. Người chơi phải tìm và click vào các số theo thứ tự trong thời gian 1 phút.

Mỗi số đúng sẽ được + 1 điểm. Số sai sẽ không được tính điểm. Số đã click rồi sẽ được tô màu khác các số còn lại. Khi hết giờ hiện ra màn hình thông báo số điểm.

Bài tập

8. Gallery hình ảnh

Bài tập

9. Làm 1 ứng dụng chat đơn giản như facebook, khi nhập text, click Send, đoạn text sẽ hiện lên trên khung chat.

** Có thể nâng cao cho chat giữa 2 users bằng cách thêm 1 textbox cho user thứ 2. Chat của user 1 sẽ được canh trái, chat của user 2 sẽ được canh phải.

Bài tập

10. Trò chơi cho phép người chơi trang trí khu vườn của mình bằng cách mua các chậu hoa với các mức tiền khác nhau. Người chơi ban đầu có sẵn 1800 đồng tiền.

Bước 1: Người chơi click chọn một trong 3 chậu hoa mình mong muốn mua.

Bước 2: Người chơi click vào các dĩa mình mong muốn đặt chậu hoa. Khi người chơi click vào dĩa:

- Nếu người chơi chưa thực hiện Bước 1 (chưa chọn chậu hoa nào) --> Hiển thị thông báo "Bạn phải chọn một chậu hoa".

- Nếu số tiền chậu hoa đang chọn lớn hơn số tiền người dùng có --> Hiển thị thông báo "Bạn không đủ tiền mua chậu hoa này".

- Nếu click vào dĩa đã có chậu hoa --> Hiển thị thông báo "Không thể đặt chậu hoa ở đây".

- Nếu ngoài khai điều kiện trên, trò chơi sẽ thực hiện các việc sau:

+ Hiển thị hình ảnh chậu hoa đã chọn ở dĩa tương ứng (dùng class ".flower" để hiển thị hình ảnh chậu hoa).

+ Trừ đi số tiền chậu hoa đã mua vào số tiền đang có.

- Khi click chọn chậu hoa, chậu được chọn sẽ được tô một viền vàng, khi đổi qua chậu khác, chậu cũ sẽ được bỏ viền vàng

Thanks for your attention!

FACULTY OF
INFORMATION TECHNOLOGY
THU DUC COLLEGE OF TECHNOLOGY

Phone: (+848) 22 158 642
Email: fit@tdc.edu.vn
Website: fit.tdc.edu.vn
Facebook: facebook.com/tdc.fit
Youtube: youtube.com/fit-tdc