




TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

ỨNG DỤNG WEB

Nguyễn Thị Mai Trang

1




TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Mục tiêu

- Trình bày được ý nghĩa của ngôn ngữ Javascript trong xử lý trang web.
- Sử dụng được cú pháp của ngôn ngữ kịch bản JavaScript để lập trình xử lý trang HTML
- Lập trình xử lý được các sự kiện cơ bản với JavaScript.
- Kết hợp được JavaScript với HTML5 để vẽ các đối tượng đồ họa 2D trên trang web.

3




TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Chương 4

Ngôn ngữ Javascript

2



TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Nội dung

- Giới thiệu JavaScript
- Ngôn ngữ kịch bản JavaScript
- Các đối tượng trong JavaScript
- Xử lý sự kiện
- Kiểm chứng dữ liệu
- Giới thiệu về đồ họa với JavaScript và HTML5

4

4.1. Giới thiệu JavaScript

- JavaScript là một ngôn ngữ lập trình web phía client, cho phép:
 - Truy xuất các phần tử trong trang web
 - Thêm các phần tử mới vào trang web.
 - Cập nhật, thay đổi nội dung trang web.
 - Cập nhật, thay đổi thuộc tính và style của các phần tử trên trang web.
 - Thực hiện các phép tính toán học đối với dữ liệu trên trang web.
 - Xử lý sự kiện khi người dùng tương tác lên một phần tử trên trang web.
 - Truy xuất ngày và giờ hiện tại từ máy tính của người dùng.
 - Xác định kích thước màn hình, phiên bản trình duyệt, độ phân giải màn hình.
 - Kiểm tra dữ liệu nhập của người dùng, cảnh báo nếu dữ liệu không hợp lệ.
 - ...

5

Giới thiệu JavaScript (tt)

- Nhúng Javascript vào trang web:
 - Đặt giữa cặp thẻ `<script></script>`
 - Cú pháp khai báo:


```
<script type= "text/javascript">
//mã lệnh JS
</script >
```

Hoặc

```
<script>
//mã lệnh JS
</script >
```
- Vị trí đặt mã lệnh JS:
 - Đặt trong thẻ `<head> </head>`
 - Đặt trong thẻ `<body> </body>`
 - Đặt trong tập tin riêng có phần mở rộng là JS.

7

Giới thiệu JavaScript (tt)

- Lợi ích của Javascript:
 - Xử lý đồ họa:
 - Kết hợp HTML5 vẽ các đối tượng đồ họa, thay thế cho các công nghệ cũ như Flash, Gif, Java Applet,...
 - Vẽ các loại biểu đồ với Chart.js, D3.js, Google Chart, Plotly
 - Là kiến thức nền tảng để phát triển web sử dụng các đối tượng và các công nghệ hiện đại
 - Các đối tượng JSON
 - Tương tác với server: AJAX
 - ReactJS, React Native, Web API
 - ...

6

Giới thiệu JavaScript (tt)

- Đặt trong `<head> </head>`:** thường là các hàm (function), chỉ thực thi khi có lời gọi hàm.

```
<head>
<script>
function Tong(a, b) {
  document.getElementById("kq").innerHTML= a + b;
}
</script>
</head>
<body>
<p id="kq">Kết quả phép cộng 8 + 6</p>
<button type="button" onclick="Tong(8,6)">Tính tổng</button>
</body>
```

Khi trang web mới load
Kết quả phép cộng 8 + 6

Tính tổng

Sau khi click vào nút Tính tổng
14

Tính tổng

8

Giới thiệu JavaScript (tt)

- **Đặt trong <body> </body>**: mã lệnh JS không nằm trong hàm sẽ thực thi tự động khi trang web được nạp theo thứ tự từ trên xuống

```
<body>
<p id="kq"></p>
<script>
  document.getElementById("kq").innerHTML= 8 + 6;
</script>
</body>
```

9

Giới thiệu JavaScript (tt)

- **Đặt trong tập tin JS (tt) – ví dụ**

```
//Nội dung tập tin html
<head>
<script src="ex_js.js"></script>
</head>
<body>
<p id="kq">Kết quả phép cộng</p>
<button type="button" onclick="Tong(10, 5)">Tính tổng</button>
</body>

//Nội dung tập tin ex_js.js
function Tong(a, b) {
  document.getElementById("kq").innerHTML= a + b;
}
```

Giao diện ban đầu

Kết quả phép cộng

Tính tổng

Sau khi click nút Tính tổng

15

Tính tổng

11

Giới thiệu JavaScript (tt)

- **Đặt trong tập tin JS (External JavaScript):**
 - Mã lệnh JS đặt trong tập tin riêng có phần mở rộng là .js
 - Lợi ích khi sử dụng External JavaScript
 - Tách HTML và mã JS → dễ đọc và sửa đổi
 - Cached tập tin JavaScript → tăng tốc độ tải trang
 - Khai báo JS trong phần tử <head> với cú pháp như sau:

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

10

Nội dung

- Giới thiệu JavaScript
- **Ngôn ngữ JavaScript**
- Các đối tượng trong JavaScript
- Xử lý sự kiện
- Kiểm chứng dữ liệu
- Giới thiệu về đồ họa với JavaScript và HTML5

12

4.2.1 Cú pháp JavaScript

- Tương tự các ngôn ngữ lập trình phổ biến khác (C, C++, Java):
 - Cuối mỗi câu lệnh thường có dấu chấm phẩy (bắt buộc nếu các câu lệnh nằm trên cùng một dòng)
 - Định danh dùng đặt tên cho biến, cho hàm có phân biệt chữ hoa, chữ thường.
- Chú thích trong JS:
 - Chú thích trên một dòng:
 - Cú pháp: **// văn bản chú thích**
 - Chú thích nhiều dòng:
 - Cú pháp: **/* các dòng văn bản chú thích */**
- Phần tử `<noscript></noscript>`: chứa nội dung thay thế nếu trình duyệt không hỗ trợ JavaScript

13

13

Cú pháp JavaScript (tt)

- `document.write()`, `document.writeln()`: ví dụ

```
<pre>
<script>
    document.writeln("<b>Dòng thứ nhất</b>");
</script>
</pre>
<script>
    document.write("Dòng thứ ");
    document.write("hai");
</script>
```

Dòng thứ nhất

Dòng thứ hai

15

15

Cú pháp JavaScript (tt)

- Ghi nội dung trên trang HTML (có thể ghi đè các nội dung đã có):
 - `document.write("content")`: ghi nội dung và không xuống dòng.
 - `document.writeln("content")`: ghi nội dung và xuống dòng (nên đặt mã JS trong phần tử `<pre>`)

14

14

Cú pháp JavaScript (tt)


- Hiển thị nội dung các phần tử HTML:
 - Sử dụng thuộc tính **innerHTML** của phần tử.
 - Để truy xuất một phần tử trong trang web, sử dụng phương thức **document.getElementById("element_id")**
 - `element_id` là giá trị thuộc tính id của phần tử

```
<h1 id="tieude"></h1> <p id="noidung"></p>
<script>
    document.getElementById("tieude").innerHTML="Ví dụ";
    document.getElementById("noidung").innerHTML=
        "Hiển thị nội dung trên phần tử &lt;p>";
</script>
```

Ví dụHiển thị nội dung trên phần tử `<p>`

16

16

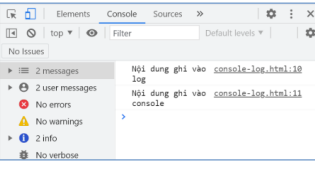


Cú pháp JavaScript (tt)

- Hiện thị nội dung trong cửa sổ console của trình duyệt:
 - Sử dụng phương thức **console.log()**, **console.info()**,...
 - Thường được dùng để kiểm tra giá trị các biến, đối tượng khi lập trình (debug).
 - Nhấn phím F12, chọn tab console.


```

<body>
<script>
  console.log("Nội dung ghi vào log");
  console.info("Nội dung ghi vào console");
</script>
</body>
      
```



17

17




Biến (tt)

- Quy tắc đặt tên biến trong JS:
 - Tên biến nên có ý nghĩa gợi nhớ và bắt đầu bằng chữ cái hoặc ký tự gạch dưới '_'.
 - Tên biến có phân biệt chữ hoa chữ thường.
- Khai báo biến:
 - Không cần xác định kiểu dữ liệu khi khai báo.
 - Có thể gán giá trị cho biến khi khai báo.
 - Khi khai báo biến mà không gán giá trị, biến sẽ có giá trị là **undefined**

19

19



4.2.2 Biến


- Có hai loại kiểu dữ liệu trong JavaScript:
 - Primitive data type

Kiểu dữ liệu	Mô tả
string	Chuỗi các ký tự liên tiếp nhau, ví dụ: "hello"
number	Kiểu số, ví dụ: 100
boolean	Kiểu luận lý, chỉ có hai giá trị là true, false
undefined	Giá trị không xác định, ví dụ một biến chưa được gán giá trị
null	Không có giá trị

- Kiểu dữ liệu tham chiếu (còn gọi là kiểu đối tượng): Object, Array, RegExp,...

18

18



Biến (tt)

- Khai báo biến với **var**
 - Ví dụ: **var number = 10;**
- Khai báo biến với **let**
 - Ví dụ: **let number = 10;**
- Khai báo hằng với **const**:
 - Ví dụ: **const number = 10;**

20

20

Biến (tt)

• Tầm vực của biến:

- Biến cục bộ (local variance): khai báo trong một hàm, chỉ có phạm vi trong hàm đó và bắt buộc phải khai báo bằng từ khóa **var**, hoặc **let**
- Biến toàn cục (global variance): khai báo ngoài các hàm, có thể truy xuất từ bất kỳ nơi đâu trong script, không nhất thiết phải khai báo bằng từ khóa **var**.
- Khi gán một giá trị cho biến không được khai báo trong hàm, biến đó được xem như biến toàn cục
- Nếu sử dụng **let/var** để khai báo, tầm vực truy xuất của biến phụ thuộc vào vị trí khai báo: ngoài hàm, trong hàm, trong khối.

21

21

Biến (tt)

• Chuyển kiểu dữ liệu

- **Number()**: chuyển đối tượng về số
- **parseInt()**: chuyển chuỗi số về số nguyên
- **parseFloat()**: chuyển chuỗi số về số thực
- **toFixed()**: chuyển số về dạng chuỗi có định dạng
- **toString()**: chuyển đối tượng về dạng chuỗi

23

23

Biến (tt)

```
<h2> Ví dụ sử dụng hai biến toàn cục x = 3, y = 5</h2>
<p id="kq"></p>
<button onClick="Cong()">Cong hai số</button>
<button onClick="Tru()">Trừ hai số</button>
```

```
<script>
```

```
var x = 3, y = 5;
function Cong() {
  var z = x + y;
  document.getElementById("kq").innerHTML = z;
}
function Tru() {
  var z = x - y;
  document.getElementById("kq").innerHTML = z;
}
```

```
</script>
```

Ví dụ sử dụng hai biến toàn cục x = 3, y = 5

8 Kết quả khi click nút Cộng hai số

Cộng hai số Trừ hai số

Ví dụ sử dụng hai biến toàn cục x = 3, y = 5

-2 Kết quả khi click nút Trừ hai số

Cộng hai số Trừ hai số

22

22


Biến (tt)

- **Number()**: chuyển biến/đối tượng sang kiểu số, nếu đối số không phù hợp → trả về NaN.

```
var a = true;
var a1 = Number(a); // a1 = 1
var b = false;
var b1 = Number(b); // b1 = 0
var str1 = 'abc';
var c = Number(str1); // c = NaN
var str2 = '100';
var d = Number(str2); // d = 100
```

24

24


 **TRƯỜNG ĐẠI HỌC MỞ TP. HCM**
Cơ hội học tập cho mọi người

Biến (tt)

- **parseInt(), parseFloat():** chuyển chuỗi số về kiểu số nguyên, số thực. nếu đối số không phù hợp → trả về NaN.

```
var a = true;
var a1 = parseInt(a); // a1 = NaN
var str1 = '10.3 xyz';
var b = parseInt(str1); // b = 10
var c = parseFloat(str1); // c = 10.3
var str2 = '100.55';
var d = parseInt(str2); // d = 100
var e = parseFloat(str2); // e = 100.55
```

25

 **TRƯỜNG ĐẠI HỌC MỞ TP. HCM**
Cơ hội học tập cho mọi người

Biến (tt)


- **toString():**

```
var x = 123;
typeof x; // → number
x = x.toString();
typeof x; // → string
```

- **toString(radix):** chuyển sang chuỗi có cơ số là radix (2 - 36)

```
let n = 10;
console.log(n.toString(2)); // in 10 ở dạng nhị phân
// "1010"
```

27

 **TRƯỜNG ĐẠI HỌC MỞ TP. HCM**
Cơ hội học tập cho mọi người


Biến (tt)

- **toFixed(n):** chuyển n về chuỗi số có định dạng xuất.

```
var x = 7.67;
x.toFixed(0); // → 7
x.toFixed(4); // → 7.6700
```

- **toString():** chuyển sang kiểu String.
 - **toString()**
 - **toString(radix)**

26

 **TRƯỜNG ĐẠI HỌC MỞ TP. HCM**
Cơ hội học tập cho mọi người

4.2.2 Toán tử trong Javascript

- Toán tử số học

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

28

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Toán tử trong Javascript (tt)

- Toán tử tăng, giảm, 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

29

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Toán tử trong Javascript (tt)

- Toán tử đối với chuỗi

- +: Nối 2 chuỗi

```
txt1 = "John";
txt2 = "Doe";
txt3 = txt1 + " " + txt2;
```

- +=: Nối chuỗi vào một chuỗi

```
txt1 = "What a very ";
txt1 += "nice day";
```

- Cộng chuỗi và số: kết quả là chuỗi

```
z = "Hello" + 5 + 3; → z = "Hello53"
```

- Trường hợp trước chuỗi là biểu thức, kết quả là trị của biểu thức + chuỗi

```
var x = 16 + 4 + "Volvo"; → x = 20Volvo"
```

31

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Toán tử trong Javascript (tt)

- Toán tử so sánh

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

30

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

4.2.4 Các hộp thoại trong JS

- Hộp thoại hiển thị cảnh báo cho người dùng: alert()

```
alert("Nội dung thông báo");
```

- Hộp thoại nhận giá trị nhập

- prompt ("chuỗi thông báo")
- prompt ("chuỗi thông báo", "giá trị gợi ý")

```
var number = prompt("Hãy nhập một số nguyên!", 10);
alert("Bạn vừa nhập số " + number);
```

32

Các hộp thoại trong JS (tt)

- Hộp thoại xác nhận thao tác: confirm()**

```
<head>
<script>
function Ketthuc() {
  if (confirm("Bạn muốn đóng trang web này?" == true)
    window.close();
}
</script>
</head>
<body>
<button onclick="Ketthuc()"> Đóng trang web </button>
</body>
```



33

Hàm (tt)

- Gọi hàm**
 - Cú pháp: Tên hàm (danh sách đối số)
 - Trường hợp hàm có trả trị, có thể gán hàm cho biến

```
<script>
function Tinh tong (a, b) {
  return a + b;
}
let a= prompt("Nhập số thứ nhất");
let b= prompt("Nhập số thứ hai");
let kq = Tinh tong(Number(a), Number(b));
alert("Tổng hai số vừa nhập là: " + kq);
</script>
```

35

4.2.5 Hàm

- Hàm trong Javascript được khai báo với từ khóa **function**
- Cú pháp:


```
function name(parameter1, parameter2, parameter3) {
  //code to be executed
}
```
- Hàm có trả trị: cuối hàm có lệnh return


```
function myFunction(p1, p2) {
  return p1 * p2;
}
```

34


Hàm (tt)

- Gọi hàm**
 - Lưu ý:
 - JS không bắt lỗi cú pháp khi không truyền đúng đối số nhưng kết quả có thể bị sai, vì giá trị của đối số không truyền vào = **undefined**
 - Khi gọi hàm thiếu (): trả về đối tượng hàm

```
<p id="demo"></p>
<script>
function toCelsius(f) {
  return (5/9) * (f-32);
}
document.getElementById("demo").innerHTML = toCelsius;
</script>
```

Kết quả
function toCelsius(f) { return (5/9) * (f-32); }

36



TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Hàm (tt)

- Một số hàm cơ bản trong JS
 - eval (exp): đánh giá biểu thức chuỗi exp và thi hành như mã JS


```
var exp = "if (3>2) document.Write ('true');" +
            " else document.write ('false');";
var result = eval (exp); // "true"
var x = eval ("2+2"); //x=4
```
 - isNaN (value): kiểm tra giá trị value
 - value là số: trả về false
 - value không phải số: trả về true

37


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người


Cấu trúc điều kiện

- Cấu trúc if
 - Cú pháp:


```
if (condition) {
  //code thực thi nếu condition = true
}
```
- Cấu trúc if .. else
 - Cú pháp:


```
if (condition) {
  //code thực thi nếu condition = true
} else {
  //code thực thi nếu condition = false
}
```


39


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

4.2.6 Cấu trúc điều kiện

- Cấu trúc lựa chọn (if..else)
- Cấu trúc rẽ nhánh (switch)
- Cấu trúc lặp
 - for
 - while
 - do..while
- Lệnh break, continue

38


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Cấu trúc điều kiện (tt)

- Cấu trúc if .. else if
 - Cú pháp:


```
if (condition1) {
  //code thực thi nếu condition1 = true
} else if (condition2) {
  //code thực thi nếu condition1 = false
  và condition2 = true
} else {
  //code thực thi nếu condition1
  = false và condition2 = false
}
```

40

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Cấu trúc rẽ nhánh

- **Cấu trúc switch**
 - Cú pháp:


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

```
switch (new Date().getDay()) {
    case 6:
        text = "Today is Saturday";
        break;
    case 0:
        text = "Today is Sunday";
        break;
    default:
        text = "working day ";
}
```

41

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Cấu trúc lặp (tt)

- **Cấu trúc lặp for/in:** lặp qua các thuộc tính của đối tượng
 - Cú pháp:


```
for (attr in obj) {
    //process obj [attr]
}
```

```
<script>
var str = "";
var sinhvien = { MSSV:"SV01",
                  Hoten:"Nguyễn Văn Nam",
                  Namsinh:2000
};
var x;
for (x in sinhvien)
    str += sinhvien[x] + " ";
alert(str);
</script>
```

This page says
SV01 Nguyễn Văn Nam 2000

OK

43

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Cấu trúc lặp

- **Cấu trúc lặp for**
 - Cú pháp:


```
for (statement 1; statement 2; statement 3) {
    code block to be executed
}
```

```
<script>
var cars = ["Vios", "Innova", "Honda City", "Ford"];
var text = "";
var i;
for (i = 0; i < cars.length; i++) {
    text += cars[i] + "<br>";
}
alert(text);
</script>
```

This page says
Vios
Innova
Honda City
Ford

OK

42

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Cấu trúc lặp (tt)


- **Cấu trúc lặp while:**
 - Cú pháp:


```
while (condition) {
    //code block to be executed
}
```

```
function myFunction() {
    var text = "";
    var i = 0;
    while (i < 10) {
        text += "<br>The number is " + i;
        i++;
    }
    return text;
}
```

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

44


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Cấu trúc lặp (tt)


- Cấu trúc lặp do..while:**
 - Cú pháp:


```
do {
    //code block to be executed
} while (condition);
```

```
function myFunction() {
  var text = ""
  var i = 0;
  do {
    text += "<br>The number is " + i;
    i++;
  } while (i < 10);
  return text;
}
```

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

45



TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Cấu trúc lặp (tt)

- break, continue**
 - Ví dụ


```
<script>
var sum = 0, i = 0;
while (true)
{
    i++;
    if(i % 10 == 0)
        break;
    if (i % 2 == 0)
        continue;
    sum += i;
}
console.log (sum);
</script>
```


47


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Cấu trúc lặp (tt)

- Lệnh break**
 - dùng trong switch
 - dùng trong vòng lặp: thoát khỏi vòng lặp
- Lệnh continue**
 - dùng trong vòng lặp
 - khi gặp lệnh này, mã lệnh sau continue không được thực thi mà quay lên thực hiện bước lặp tiếp theo


46


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Nội dung

- Giới thiệu JavaScript
- Ngôn ngữ JavaScript
- Các đối tượng trong JavaScript**
- Xử lý sự kiện
- Kiểm chứng dữ liệu
- Giới thiệu về đồ họa với JavaScript và HTML5


48



4.3 Các đối tượng trong JavaScript

- Kiểu dữ liệu đối tượng
- String
- Date
- Math
- Array
- Mô hình BOM (Browser Object Model)
- Mô hình DOM (Document Object Model)

49



4.3.1 Kiểu dữ liệu đối tượng (object)

- Tạo đối tượng:
 - Khai báo và khởi tạo một đối tượng
 - Sử dụng phương thức khởi tạo (Constructor)
- Truy xuất thuộc tính của đối tượng theo cú pháp:
 - TenDoituong.thuoc tinh
 - Hoặc TenDoituong["thuoc tinh"]

51




4.3.1 Kiểu dữ liệu đối tượng (object)

- Ví dụ về đối tượng

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

50



Kiểu dữ liệu đối tượng (tt)


- Tạo đối tượng bằng cách khai báo và khởi tạo
 - Cú pháp:


```
var objectname = {att1:value, att2:value,...};
```

 - att1, att2: thuộc tính của đối tượng
 - value: giá trị của mỗi thuộc tính
 - Ví dụ:


```
var Rectangle = {Left: 0, Right: 100, Top: 0, Bottom:80};
var left = Rectangle.Left; // hoặc Rectangle["Left"]
var right = Rectangle.Right; // hoặc Rectangle["Right"]
var top = Rectangle.Top; // hoặc Rectangle["Top"]
var bottom = Rectangle.Bottom; //hoặc Rectangle["Bottom"]
```

52



TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Kiểu dữ liệu đối tượng (tt)


- Sử dụng phương thức khởi tạo (Constructor)
 - Tạo một đối tượng bằng cách khai báo một function
 - Tên đối tượng được viết hoa ký tự đầu.

```
function Rectangle(left, right, top, bottom) {
    this.Left = left;
    this.Right = right;
    this.Top = top;
    this.Bottom = bottom;
    this.Area = function() {return (this.Right - this.Left) * (this.Bottom - this.Top);}
}
```

- Tạo một thể hiện của đối tượng với **new**

```
var rect = new Rectangle (0,100,0,50);
var rect2 = new Rectangle (60,120,100,150);
```

53




TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

4.3.2 String

- Đối tượng String trong JS được định nghĩa để cho phép xử lý chuỗi văn bản.
- Là dãy các ký tự liên tiếp nhau, chỉ số bắt đầu là 0
- Chuỗi được đặt trong cặp nháy đơn hoặc nháy kép
- Ví dụ:


```
var mySite= "cunghoclaptrinh.com";
username = 'admin';
```

55



TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Kiểu dữ liệu đối tượng (tt)

- Sử dụng phương thức khởi tạo (tt): ví dụ

```
<p id="info"></p> <button onClick="TestObject()">View Area</button>
<script>
function Rectangle(left, right, top, bottom) {
    this.Left = left;
    this.Right = right;
    this.Top = top;
    this.Bottom = bottom;
    this.Area = function() {return (this.Right - this.Left) * (this.Bottom - this.Top)}
}
function TestObject() {
    var rect1 = new Rectangle (0,100,0,50);
    var rect2 = new Rectangle (60,120,100,150);
    var area1 = rect1.Area();
    var area2 = rect2.Area();
    document.getElementById("info").innerHTML =
        "Area of rect1: "+ area1+
        "<br>Area of rect2: "+ area2;
}
</script>
```

View Area


Khi trang web được nạp

View Area

Sau khi click nút View Area

Area of rect1: 5000
Area of rect2: 3000

54



TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

String (tt)

- Có thể khai báo biến đối tượng String theo cú pháp:


```
TenBien = new String ("Chuỗi ký tự");
```


 - Ví dụ:


```
var str = new String("Ví dụ tạo đối tượng String");
```
- Có thể ngắt một chuỗi dài trong code bằng các sử dụng ký tự '\':


```
document.getElementById("demo").innerHTML = "Hello \
World!";
```
- Thuộc tính **length**: số ký tự trong một chuỗi.
 - Ví dụ:


```
userName = "Thanh";
userLength = userName.length; //userLength = 5
```

56




String (tt)

- **Một số phương thức của đối tượng String:**
 - *Lưu ý: tất cả các phương thức với String đều trả về một giá trị mới, không làm thay đổi chuỗi gốc*
 - **charAt(index):** Trả về ký tự trong chuỗi tại vị trí index. Ví dụ:

```
var str = 'JavaScript';
var ch = str.charAt(4); //ch = 'S'
```
 - **charCodeAt(index):** Trả về mã Accii của ký tự trong chuỗi tại vị trí index. Ví dụ:

```
var str = 'JavaScript';
var ascii = str.charCodeAt(1); //ascii = 97 (mã ASCII ký tự 'a')
```

57




String (tt)

- **search(regex):** tương tự như indexOf nhưng đối số regex thường là biểu thức chính quy (regular expression / RegExp)
 - RegExp: là một mẫu ký tự (pattern), được dùng trong tìm kiếm và thay thế. Cú pháp: `/pattern/modifier(s);` (tham khảo: https://www.w3schools.com/jsref/jsref_obj_regexp.asp)
 - modifier:
 - g: đối sánh và thực hiện với tất cả các kết quả phù hợp
 - i: đối sánh không phân biệt chữ hoa chữ thường
 - m: đối sánh nhiều dòng

```
//tìm ký tự a, b hoặc c không phân biệt chữ
hoa hay chữ thường trong chuỗi str
var str = "Red Blue Cyan red blue cyan";
var n = str.search(/[abc]/i); //4 (vị trí ký tự B)
```

```
//tìm chuỗi BLUE trong chuỗi str không phân
biệt chữ hoa, thường
var str = "Red Blue Cyan red blue cyan";
var token = new RegExp("BLUE", "i");
var n = str.search(token); //4 (vị trí ký tự B)
```

59




String (tt)

- **indexOf(searchValue [,from_index]):** trả về vị trí tìm thấy chuỗi searchValue lần đầu tiên trong chuỗi, với vị trí bắt đầu tìm là from_index (mặc định là 0), nếu không tìm thấy, trả về -1. Ví dụ:

```
var str = 'Hello, JavaScript! Welcome to JavaScript tutorial';
var pos1 = str.indexOf('JavaScript'); //7
var pos2 = str.indexOf('JavaScript',20); //30
```
- **lastIndexOf(searchValue [,fromIndex]):** tương tự như indexOf, nhưng hướng tìm kiếm bắt đầu từ cuối chuỗi. Ví dụ:

```
var str = 'Hello, JavaScript! Welcome to JavaScript tutorial';
var pos1 = str.lastIndexOf('JavaScript'); //30
var pos2 = str.lastIndexOf('JavaScript',20); //7
```
- **includes(searchValue, start):** trả về true nếu trong chuỗi có chứa chuỗi searchValue, ngược lại trả về false
 - start: vị trí bắt đầu tìm, tùy chọn, mặc định là 0

58



String (tt)

- **slice(start [, end]):** trả về chuỗi con của một chuỗi bắt đầu từ vị trí start đến end (không bao gồm ký tự tại vị trí end).
 - Nếu start < 0: trả về chuỗi là start ký tự cuối cùng của chuỗi
 - Nếu end < 0: trả về chuỗi lấy từ vị trí start đến hết chuỗi, loại bỏ end ký tự cuối cùng của chuỗi.
 - Nếu không có end: trả về chuỗi bắt đầu từ vị trí start đến hết chuỗi.
- Ví dụ:

```
var str = "Red Blue Cyan Yellow";
var s1 = str.slice(4,8); // Blue
var s2 = str.slice(4); // Blue Cyan Yellow
var s3 = str.slice(4,-6); // Blue Cyan
var s4 = str.slice(-6); // Yellow
```

60

String (tt)

- **substring(start [,end]):** tương tự slice, nhưng không nhận đối số âm.
- **substr(start [,length]):** trả về chuỗi con từ một chuỗi bắt đầu từ vị trí start, lấy length ký tự. Nếu không có length, chuỗi kết quả là chuỗi con từ vị trí start đến hết chuỗi.

Ví dụ:

```
var str = "Red Blue Cyan Yellow";
var s1 = str.substr(4); // Blue Cyan Yellow
var s2 = str.substr(4, 9); // Blue Cyan
```

61

61

String (tt)

- **localeCompare(compareString):** so sánh chuỗi hiện tại với chuỗi compareString, trả về các giá trị:

- -1: chuỗi gọi nhỏ hơn compareString
- 0: hai chuỗi bằng nhau
- 1: chuỗi gọi lớn hơn compareString

Ví dụ:

```
let text1 = "ab";
let text2 = "cd";
let result = text1.localeCompare(text2); // result = -1
```

- **repeat(count):** trả về chuỗi được nhân bản từ chuỗi gọi count lần

Ví dụ:

```
let text = "Hello world!";
let result = text.repeat(4);
//result = "Hello world! Hello world! Hello world! Hello world!"
```

63

63

String (tt)

- **toLowerCase(), toLocaleLowerCase():** trả về chuỗi đã được chuyển thành chữ thường.

Ví dụ:

```
• var str = "Red Blue Cyan Yellow";
• var s1 = str.toLowerCase(); // red blue cyan yellow
```

- **toUpperCase(), toLocaleUpperCase():** trả về chuỗi đã được chuyển thành chữ hoa.

Ví dụ:

```
• var str = "Red Blue Cyan Yellow";
• var s1 = str.toUpperCase(); // RED BLUE CYAN YELLOW
```

62

62

String (tt)

- **replace(searchValue, newValue):** trả về chuỗi đã được thay thế searchValue thành newValue ở vị trí tìm thấy đầu tiên

Ví dụ:

```
let text = "Visit Microsoft!Microsoft";
let result = text.replace("Microsoft", "W3Schools");
//result = Visit W3Schools!Microsoft
```

64

64

String (tt)

- **concat(string2[, string3, ..., stringN])**: trả về chuỗi là kết quả chuỗi hiện tại nối với các chuỗi string2, string3, ..., stringN.

Ví dụ:

```
var str = "Red Blue Cyan Yellow";
var s = str.concat(" Black", " White");
// Red Blue Cyan Yellow Black White
```

- **startsWith(searchString[, position])**: trả về true, nếu chuỗi bắt đầu với chuỗi searchString, tính từ vị trí position (mặc định là 0), ngược lại trả về false.

Ví dụ:

```
var str = "Red Blue Cyan Yellow";
var bStart = str.startsWith('Red'); //true
var bStart1 = str.startsWith('Blue', 4); //true
```

65

65

String (tt)

- **trim()**: trả về chuỗi đã được cắt bỏ các khoảng trắng đầu và cuối chuỗi.

Ví dụ:

```
var str = " Red ";
var s = str.trim(); // "Red"
```

- **trimStart(), trimLeft()**: trả về chuỗi đã được cắt bỏ các khoảng trắng đầu chuỗi.

Ví dụ:

```
var str = " Red ";
var s = str.trimStart(); // "Red "
```

- **trimEnd(), trimRight()**: trả về chuỗi đã được cắt bỏ các khoảng trắng cuối chuỗi.

Ví dụ:

```
var str = " Red ";
var s = str.trimEnd(); // " Red"
```

67

67

String (tt)

- **endsWith(searchString[, length])**: trả về true, nếu chuỗi kết thúc là searchString, với length là chiều dài đoạn chuỗi đang xét (mặc định là hết chuỗi), ngược lại trả về false.

Ví dụ:

```
var str = "Red Blue Cyan Yellow";
var b1 = str.endsWith("Yellow"); //true
var b2 = str.endsWith("Cyan"); //false
var b3 = str.endsWith("Cyan", 13); //true
```

66

66

String (tt)

- **split([separator[, limit]])**: tách một đối tượng String thành một mảng các chuỗi. Trong đó:

- separator là chuỗi chứa các ký tự dùng để phân cách
- limit: số ký tự trong mảng kết quả, nếu không chỉ định đối số này, mặc định là tách toàn bộ chuỗi vào mảng.

- Ví dụ:

```
var str = "a,b,c,d,e,f";
var arr1 = str.split(","); // arr1 = ['a','b','c','d','e','f']
var x1 = arr1[0]; // x1 = 'a'
var str1 = "Hello";
var arr2 = str1.split(""); // arr2 = ['H','e','l','l','o']
var x2 = arr1[1]; // x2 = 'e'
```

68

68

4.3.3 Date

- Đối tượng cho phép làm việc với ngày, giờ
- Các phương thức tạo đối tượng Date:
 - **Date()**: tạo chuỗi ngày giờ hiện tại
 - **Date(milliseconds)**: tạo chuỗi ngày giờ với đối số là số nguyên tính theo mili giây
 - **Date(dateString)**: tạo chuỗi ngày giờ với đối số là một chuỗi ngày giờ hợp lệ.
 - **Date(year, month, day, hours, minutes, seconds, milliseconds)**: tạo chuỗi ngày giờ với các đối số tương ứng theo thứ tự: năm, tháng, ngày, giờ, phút, giây.
 - Nếu các đối số phía sau không có, mặc định là 0.
 - month có giá trị từ 0 - 11 (January = 0, December = 11)

69

69

Date (tt)

- Một số phương thức của đối tượng Date:
 - **toString()**: chuyển Date thành chuỗi ngày giờ theo định dạng mặc định trên hệ thống.
 - **toUTCString()**: chuyển Date thành chuỗi dạng chuẩn.
 - **toDateString()**: chuyển Date thành chuỗi dạng ngày.
 - **toTimeString()**: chuyển Date thành chuỗi dạng giờ.
 - **parse()**: chuyển Date sang mili giây.
 - **getDate()**: trả về giá trị ngày (1 - 31).

71

71

Date (tt)

- Các phương thức tạo đối tượng Date: ví dụ

```
<body>
<p id="d1"></p> <p id="d2"></p> <p id="d3"></p> <p id="d4"></p>
<script>
  var d1 = new Date();
  document.getElementById("d1").innerHTML = d1;
  var d2 = new Date (1541330831000);
  document.getElementById("d2").innerHTML = d2;
  var d3 = new Date("October 13, 2018 11:13:00");
  document.getElementById("d3").innerHTML = d3;
  var d4 = new Date (2018,5,24,11,33,30,0);
  document.getElementById("d4").innerHTML = d4;
</script>
</body>
```

Sun Nov 04 2018 18:29:34 GMT+0700 (Indochina Time)
 Sun Nov 04 2018 18:27:11 GMT+0700 (Indochina Time)
 Sat Oct 13 2018 11:13:00 GMT+0700 (Indochina Time)
 Sun Jun 24 2018 11:33:30 GMT+0700 (Indochina Time)

70


70

Date (tt)

- Một số phương thức của đối tượng Date (tt):
 - **getDay()**: trả về thứ tự ngày trong tuần từ 0 - 6 (Sunday - Saturday)
 - **getFullYear()**: trả về giá trị năm gồm 4 số.
 - **getHours()**: trả về giá trị giờ (0 - 23).
 - **getMinutes()**: trả về giá trị phút (0 - 59).
 - **getSeconds()**: trả về giá trị giây (0 - 59).
 - **getMilliseconds()**: trả về giá trị mili giây (0 - 999).
 - **getMonth()**: trả về giá trị tháng (0 - 11).
 - **getTime()**: trả về giá trị mili giây tính từ 1/1/1970.

72


72


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Date (tt)

- Một số phương thức của đối tượng Date (tt):
 - setDate()**: thiết lập giá trị ngày (1 - 31).
 - setDay()**: thiết lập thứ tự ngày trong tuần (0 - 6).
 - setFullYear()**: thiết lập giá trị năm gồm 4 số.
 - setHours()**: thiết lập giá trị giờ (0 - 23).
 - setMinutes()**: thiết lập giá trị phút (0 - 59).
 - setSeconds()**: thiết lập giá trị giây (0 - 59).
 - setMilliseconds()**: thiết lập giá trị mili giây (0 - 999).
 - setMonth()**: thiết lập giá trị tháng (0 - 11).
 - setTime()**: thiết lập giá trị mili giây tính từ 1/1/1970.


73


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

4.3.4 Math

- Đối tượng chứa các phương thức cho phép làm việc với các phép tính số học
- Không cần khai báo và khởi tạo
- Sử dụng:
 - Math.TenPhuongThuc ()
 - Math.TenThuocTinh

75


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Date (tt)


```

<p id="d"></p>
<script>
  var d = new Date();
  var nam = d.getFullYear();
  var thang = d.getMonth() + 1 ;
  var ngay =d.getDate();
  var thu = d.getDay() + 1 ;
  var gio = d.getHours();
  var phut = d.getMinutes();
  var giay = d.getSeconds();
  document.getElementById("d").innerHTML = "Hôm nay, thứ " +
    thu + " ngày " + ngay + "/" + thang + "/" + nam +
    " " + gio + ":" + phut + ":" + giay;
</script>

```

Hôm nay, thứ 3 ngày 20/7/2021 10:18:11

74


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Math (tt)

- Một số thuộc tính của đối tượng Math:
 - Math.E: hằng số e.
 - Math.LN2: logarithm tự nhiên của 2 (~0.693).
 - Math.LN10: logarithm tự nhiên của 10 (~2.303).
 - Math.LOG2E: logarithm trên cơ số 2 của e (~1.443).
 - Math.LOG10E: logarithm trên cơ số 10 của e (~0.434).
 - Math.PI: hằng số PI (~3.14159).
 - Math.SQRT1_2: căn bậc 2 của 1/2 (~0.707).
 - Math.SQRT2: căn bậc 2 của 2 (~1.414).

76

Math (tt)

- Một số phương thức của đối tượng Math:
 - Math.abs(x): trả về trị tuyệt đối của x.
 - Math.exp(x): trả về giá trị e mũ x.
 - Math.sqrt(x): trả về căn bậc 2 của x.
 - Math.pow(x, y): trả về giá trị x mũ y.
 - Math.random(): trả về một số thực ngẫu nhiên từ 0 đến 1.
 - Math.round(x): trả về số nguyên được làm tròn từ số thực x.
 - Math.ceil(x): trả về số nguyên được làm tròn lên từ số thực x.

77

77

4.3.5 Mảng (Array)

- Trong JS, Array là một đối tượng toàn cục được định nghĩa để lưu trữ một danh sách các phần tử.
- Array trong JS là một đối tượng nên nó có thể chứa các phần tử khác kiểu dữ liệu.
- Ngoài Array, JS còn nhiều kiểu dữ liệu tập hợp khác như: Set, Map,...
- Các cách tạo mảng:


```
TenMang = [PhanTu1, PhanTu2, PhanTu3];
TenMang = new Array();
TenMang = new Array (SoPhanTu);
TenMang = new Array (PhanTu1, PhanTu2, PhanTu3,...);
```

79

79

Math (tt)

- Một số phương thức của đối tượng Math (tt):
 - Math.floor(x): trả về số nguyên được làm tròn xuống từ số thực x.
 - Math.log(x): trả về log trên cơ số e của x.
 - Math.log10(x): trả về log trên cơ số 10 của x.
 - Math.log2(x): trả về log trên cơ số 2 của x.
 - Math.max([x[, y[, ...]]]): trả về giá trị lớn nhất của các số trong dãy các đối số x, y,...
 - Math.min([x[, y[, ...]]]): trả về giá trị nhỏ nhất của các số trong dãy các đối số x, y,...

78

78

Mảng (tt)

Ví dụ tạo mảng:

```
arrInt = new Array();
arrInt[0] = 10;
arrInt[1] = 15;
arrInt[2] = 20;

arrColor = new Array(3);
arrColor[0] = "Blue";
arrColor[1] = "Red";
arrColor[2] = "Yellow";
```

```
arrSize = new Array ("Small", "Medium", "Large");
```

Truy cập phần tử trong mảng:

Sử dụng chỉ số, phần tử đầu tiên có chỉ số là 0

Ví dụ:

```
var number = arrInt [0]; //number = 10
var color = arrColor [1]; // color = "Red"
var size = arrSize[2]; //size = "Large"
```

80

80

Mảng (tt)

- Một số thuộc tính và phương thức của đối tượng Array:

- **length**: thuộc tính trả về số phần tử trong mảng.
- **valueOf()**, **toString()**: trả về chuỗi chứa các phần tử trong mảng, các phần tử phân cách nhau bởi dấu phẩy (',').

Ví dụ:

```
arrColor = new Array("Blue", "Red", "Yellow", "Cyan");
var s1 = arrColor.valueOf(); //s1= "Blue,Red,Yellow,Cyan"
var s2 = arrColor.toString(); //s1= "Blue,Red,Yellow,Cyan"
```

81

81

Mảng (tt)

- **unshift(element1[, ...[,elementN]])**: thêm các phần tử element1,... elementN vào đầu mảng, trả về số phần tử của mảng sau khi thêm.

Ví dụ:

```
arrColor = ["Blue", "Red", "Yellow", "Cyan"];
var count = arrColor.unshift("White", "Black");
var s = arrColor.toString();
//s = "White,Black,Blue,Red,Yellow,Cyan", count = 6
```

- **shift()**: trả về giá trị phần tử đầu mảng và xóa phần tử này ra khỏi mảng, nếu mảng rỗng, trả về undefined.

Ví dụ:

```
arrColor = ["Blue", "Red", "Yellow", "Cyan"];
var color = arrColor.shift();
var s = arrColor.valueOf(); //color = "Blue", s = "Red,Yellow,Cyan"
```

83

83

Mảng (tt)

- **Array.join([separator])**: chuyển mảng thành chuỗi.

Ví dụ:

```
arrColor = new Array("Blue", "Red", "Yellow", "Cyan");
var s = arrColor.join("-"); //s= "Blue-Red-Yellow-Cyan"
```

- **reverse()**: đảo các phần tử trong mảng.

Ví dụ:

- arrInt = [1,2,3,4,5];
- arrInt.reverse(); //các phần tử trong arrInt lần lượt là 5,4,3,2,1

82

82

Mảng (tt)

- **push(element1[, ...[, elementN]])**: thêm các phần tử element1,... elementN vào cuối mảng, trả về số phần tử của mảng sau khi thêm

Ví dụ:

```
arrColor = ["Blue", "Red", "Yellow", "Cyan"];
var count = arrColor.push("White", "Black");
var s = arrColor.toString();
//s = "Blue,Red,Yellow,Cyan,White,Black", count = 6
```

- **pop()**: trả về giá trị phần tử cuối mảng và xóa phần tử này ra khỏi mảng, nếu mảng rỗng, trả về undefined.

Ví dụ:

```
arrColor = ["Blue", "Red", "Yellow", "Cyan"];
var color = arrColor.pop();
var s = arrColor.valueOf(); //color = "Cyan", s = "Blue,Red,Yellow"
```

84

84

Mảng (tt)

- **indexOf(element [,from_Index]):** trả về vị trí tìm thấy phần tử element lần đầu tiên trong mảng, với vị trí bắt đầu tìm là from_index (mặc định là 0), nếu không tìm thấy, trả về -1.
Ví dụ:
 • arrInt = [1,2,3,4,2,1,5];
 • var pos1 = arrInt.indexOf(2); //1
 • var pos2 = arrInt.indexOf(2,2); //4
- **lastIndexOf(element [,from_Index]):** tương tự như indexOf, nhưng hướng tìm kiếm bắt đầu từ cuối mảng.
Ví dụ:
 • arrInt = [1,2,3,4,2,1,5];
 • var pos1 = arrInt.lastIndexOf(1); //5
 • var pos2 = arrInt.lastIndexOf(1,3); //0

85

Mảng (tt)

- **forEach():** ví dụ

Các phần tử trong mảng: 1,2,3,4,5
 Tổng các phần tử: 15
 Tích các phần tử trong mảng: 120

```

<p id="kq"></p>
<script>
  function TinhTong(element,index, arr) {
    tong += element;
    tich *= arr[index];
  }
  arrInt = [1,2,3,4,5];
  tong = 0, tich = 1;
  arrInt.forEach(TinhTong);
  document.getElementById("kq").innerHTML=
    "Các phần tử trong mảng: " + arrInt.toString() +
    "<br>Tổng các phần tử: " + tong +
    "<br>Tích các phần tử trong mảng: " + tich;
</script>
  
```

87

Mảng (tt)

- **forEach():** duyệt và xử lý từng phần tử mảng trong một hàm callback được định nghĩa sẵn. Cú pháp:

```

forEach(function callback(currentValue[, index[, array]]) {
  //code
}, thisArg);
  
```

- Hàm callback có thể có ba tham số:
 - currentValue: giá trị phần tử đang xét
 - index: vị trí phần tử đang xét
 - array: tên mảng
 - thisArg: tùy chọn, nếu được truyền vào thì thisArg sẽ được sử dụng làm tham chiếu "this".

86

Mảng (tt)

- **sort([compareFunction]):**
 - sắp xếp các phần tử mảng theo thứ tự tăng dần
 - compareFunction: là một hàm tùy chọn quy định thứ tự sắp xếp, Nếu bỏ qua, mảng sẽ được sắp xếp theo thứ tự tăng dần.

```

<p id="p1"></p> <p id="p2"></p>
<script>
  function compare(a, b) {
    return b - a;
  }
  arrInt = [1,2,3,4,3,5];
  arrInt.sort();
  document.getElementById("p1").innerHTML = arrInt;
  arrInt.sort(compare);
  document.getElementById("p2").innerHTML = arrInt;
</script>
  
```

1,2,3,3,4,5
 5,4,3,3,2,1

88

4.3.6 Mô hình BOM (Browser Object Model)

- Các đối tượng tương tác với cửa sổ trình duyệt.
- BOM được tổ chức theo dạng phân cấp.

```

graph LR
    window --> document
    window --> frames
    window --> history
    window --> location
    window --> navigator
    window --> screen
    document --> anchors
    document --> forms
    document --> images
    document --> links
    document --> location
  
```

89

Đối tượng window (tt)

- Thuộc tính **innerWidth**: chiều rộng của cửa sổ trình duyệt.
- Thuộc tính **innerHeight**: chiều cao của cửa sổ trình duyệt.
 - Ví dụ:


```
var width = window.innerWidth;
var height = window.innerHeight;
```

91

Đối tượng window

- Đối tượng cửa sổ trình duyệt.
- Tất cả các đối tượng, hàm và biến toàn cục trong JS đều là thành viên của đối tượng window, được truy xuất thông qua đối tượng window.
- Đối tượng document cũng là một thuộc tính của đối tượng window. Để sử dụng đối tượng document, phải truy cập thông qua đối tượng window như sau:
 - `window.document.getElementById("header");`
 - Viết ngắn gọn lại: `document.getElementById("header");`

90

Đối tượng window (tt)

- Các phương thức:
 - window.open(url, name, specs, replace)**
 - url: địa chỉ trang được mở, nếu không có, mở trang trắng.
 - name: là các giá trị `_blank` (mặc định), `_parent`, `_self`, `_top`.
 - replace: tùy chọn, có tạo một mục mới trong history hay không.
 - true: url mới sẽ thay thế tài liệu hiện tại trong history
 - false: tạo mục mới trong history.

92

Đối tượng window (tt)

- specs: tùy chọn, danh sách các mục được phân cách bằng dấu phẩy, không có khoảng trắng, gồm có:
 - fullscreen: yes/no, cửa sổ trình duyệt lấp đầy/không lấp đầy màn hình.
 - width, height: chiều rộng, chiều cao cửa sổ theo pixels.
 - top, left : vị trí góc trên, trái của cửa sổ theo pixels.
 - menubar: yes/no, có hoặc không có menu.
 - resizable: yes/no, có hay không cho phép thay đổi kích thước cửa sổ.
 - scrollbars: yes/no, có hoặc không có thanh cuộn.
 - status: yes/no, có hoặc không có thanh trạng thái.
 - titlebar: yes/no, có hoặc không có thanh tiêu đề.
 - toolbar: yes/no, có hoặc không có thanh công cụ.

93

93

Đối tượng window (tt)

- **window.close()**: đóng cửa sổ trình duyệt
- **setInterval(func, time)**: cài đặt bộ định thời (timer), sau mỗi khoảng thời gian là time mili giây sẽ thực thi hàm func.
- **clearInterval(ID)**: xóa timer tạo bởi setInterval.
- **setTimeout(func, time)**: cài đặt bộ định thời (timer), sau khoảng thời gian là time mili giây sẽ thực thi hàm func nhưng chỉ thực thi một lần.
- **clearTimeout(ID)**: xóa timer tạo bởi setTimeout.

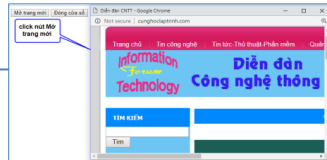
95

95

Đối tượng window (tt)

– window.open: ví dụ

```
<button onclick="openNew()">Mở trang mới</button>
<button onclick="closeMe()">Đóng cửa sổ</button>
<script>
  function openNew() {
    window.open("http://cunghoclaptrinh.com", "_blank", "toolbar=yes, scrollbars=yes,
      resizable=yes, top=100, left=200, width=640, height=480");
  }
  function closeMe() { window.close(); }
</script>
```



94

94


Đối tượng window (tt)

• Ví dụ: tạo một đồng hồ điện tử và hai nút Chạy - Dừng

```
<h1 id="dongho"></h1>
<button onclick="dongho=setInterval(startTimer, 1000)">Chạy </button>
<button onclick="clearInterval(dongho)">Dừng</button>
<script>
  dongho=setInterval(startTimer, 1000);
  function startTimer () {
    var d = new Date();
    var gio = d.getHours();
    var phut = d.getMinutes() < 10 ? "0" + d.getMinutes() : d.getMinutes();
    var giay = d.getSeconds() < 10 ? "0" + d.getSeconds() : d.getSeconds();
    document.getElementById("dongho").innerHTML = gio + ":" + phut + ":" + giay;
  }
</script>
```

96

96


 **Đối tượng window.screen**

- Đối tượng chứa thông tin về màn hình của trình duyệt người dùng.
- Các thuộc tính:
 - **screen.width**: chiều rộng của màn hình thiết bị.
 - **screen.height**: chiều cao của màn hình thiết bị.

```
<p id="info"></p>
<script>
  document.getElementById("info").innerHTML =
    "Chiều rộng màn hình là:" + window.screen.width +
    "<br>Chiều cao màn hình là:" + window.screen.height;
</script>
```

Chiều rộng màn hình là:1366
Chiều cao màn hình là:768

97


 **Đối tượng window.location (tt)**

- Ví dụ

```
<p id="info"></p>
<script>
  var info = 'host: ' + window.location.host + '<br>';
  info += 'href: ' + window.location.href + '<br>';
  info += 'origin: ' + window.location.origin + '<br>';
  info += 'pathname: ' + window.location.pathname + '<br>';
  info += 'protocol: ' + window.location.protocol + '<br>';
  document.getElementById("info").innerHTML = info;
</script>
```


hostname: cunghoclaptrinh.com
href: http://cunghoclaptrinh.com/window.location.html
origin: http://cunghoclaptrinh.com
pathname: /window.location.html
protocol: http:

99

 **Đối tượng window.location**

- Đối tượng xử lý, điều hướng url của trang web.
- Một số thuộc tính:
 - host: tên máy chủ (hostname) và port của trang web.
 - hostname: tên máy chủ của trang web.
 - href: URL của trang web.
 - origin: trả về URL của web site.
 - pathname: đường dẫn tương đối của trang.
 - search: phần query string của URL (phần sau dấu ?).
 - protocol: giao thức được sử dụng của trang web

98

 **Đối tượng window.location (tt)**

- Một số phương thức của đối tượng window.location:
 - **window.location.reload()**: nạp lại trang web (~ nhấn F5)
 - **window.location.assign(url)**: load một trang mới.
Ví dụ:
window.location.assign("http://cunghoclaptrinh.com");
 - **window.location.replace(url)**: chuyển đến trang mới, không lưu vào history.
Ví dụ:
window.location.replace("http://cunghoclaptrinh.com");

100

Đối tượng window.history

- Đối tượng lưu trữ lịch sử duyệt web.
- Hai phương thức chính:
 - **history.back()**: quay lại trang trước
 - **history.forward()**: đi đến trang kế tiếp đã duyệt qua

101

101

Đối tượng document

- Thuộc về cả hai mô hình BOM và DOM
- Được dùng để truy cập các phần tử HTML để:
 - Thay đổi nội dung các phần tử HTML.
 - Thay đổi thuộc tính các phần tử HTML.
 - Thay đổi style CSS của các phần tử HTML.
 - Thêm phần tử HTML và các thuộc tính của phần tử.
 - Xóa các phần tử HTML và các thuộc tính của phần tử.
 - Cho phép xử lý các sự kiện trong trang HTML.
 - Tạo sự kiện mới trong trang HTML.

103

103

Đối tượng window.navigator

- Đối tượng chứa thông tin về trình duyệt của người dùng.
- Một số thuộc tính:
 - **navigator.appName**: tên trình duyệt.
 - **navigator.platform**: hệ điều hành (Ví dụ: Win32).
 - **navigator.cookieEnabled**: true/false - kiểm tra cookie có/không được cho phép.
 - **navigator.appVersion**: phiên bản trình duyệt.
 - **navigator.language**: ngôn ngữ của trình duyệt.

102

102

Đối tượng document (tt)

- Một số thuộc tính của đối tượng document:
 - **document.anchors**: danh sách các phần tử <a> có thuộc tính name.
 - **document.domain**: tên domain của trang web.
 - **document.body**: phần tử <body> .
 - **document.cookie**: cookie của tài liệu.
 - **document.head**: phần tử <head>.
 - **document.links**: danh sách tất cả các phần tử <a> và các phần tử <a> có thuộc tính href.
 - **document.title**: phần tử <title>.

104

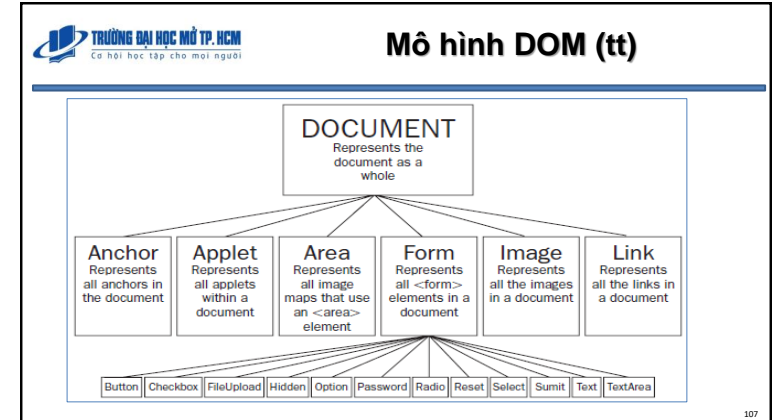
104

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Đối tượng document (tt)

- Một số thuộc tính (tt)
 - document.embeds**: danh sách tất cả các phần tử <embed>.
 - document.forms**: danh sách các phần tử <form>.
 - document.images**: danh sách các phần tử .
 - document.lastModified**: ngày giờ cập nhật tài liệu lần cuối.
 - document.inputEncoding**: encoding (character set) của tài liệu, ví dụ: UTF-8.
 - document.URL**: URL của trang web.
 - document.documentElement**: phần tử <html>.

105

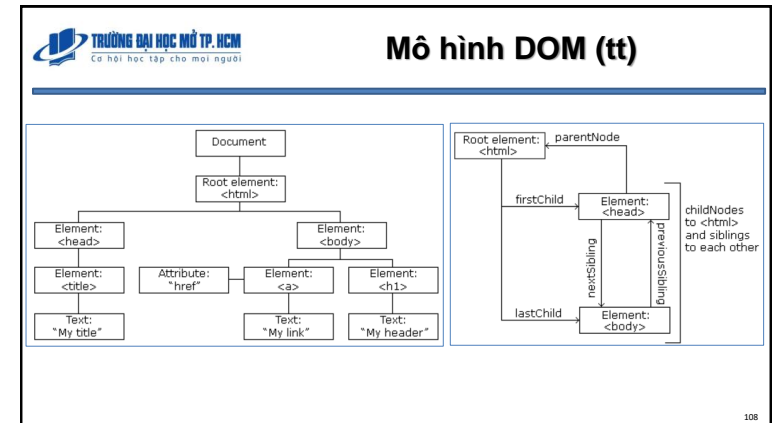


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

4.3.7 Mô hình DOM (Document Object Model)

- DOM cho phép truy cập vào HTML, thao tác và xem HTML dưới dạng tài liệu XML (eXtensible Markup Language - Ngôn ngữ đánh dấu mở rộng).
- DOM phản ánh toàn bộ cấu trúc tài liệu HTML như một cây logic, trong đó mỗi phần tử là một nút (node)
- Mỗi nút chứa dữ liệu, có thể chứa các nút con, có thể có trình xử lý sự kiện.
- Các phương thức của đối tượng document cho phép truy cập đến từng nút trong cây để có thể thay đổi cấu trúc, kiểu hoặc nội dung của phần tử.

106



Đối tượng document

- Đại diện cho toàn bộ trang HTML
- Để truy cập vào một phần tử HTML, ta cần phải truy cập đối tượng document trước
- Truy cập các phần tử HTML:**
 - `getElementById("id")`: trả về phần tử có thuộc tính id= "id"
 - `getElementsByName("element_name")`: danh sách các phần tử có thuộc tính name = "element_name".
 - `getElementsByTagName("tag_name")`: danh sách các phần tử "tag_name".

109

109

Đối tượng document (tt)

- Cập nhật nội dung phần tử trong DOM:** sử dụng đối tượng document truy cập phần tử và cập nhật thuộc tính innerHTML

```
<h1 id="header">Old Header</h1>
<script>
  var element = document.getElementById("header");
  element.innerHTML = "New Header";
</script>
```

111

111

Đối tượng document (tt)

- Truy cập các phần tử HTML (tt):**
 - `getElementsByClassName("class_name")`: danh sách các phần tử có thuộc tính class="class_name"
 - `querySelector (CSS selectors)`: chọn các phần tử theo bộ chọn CSS selectors, trả về phần tử đầu tiên được tìm thấy, hoặc null.
 - `querySelectorAll(CSS selectors)`: tương tự querySelector nhưng trả về tất cả các phần tử được tìm thấy hoặc trả về một danh sách rỗng.

110

110

Đối tượng document (tt)

- Thay đổi thuộc tính phần tử:** sử dụng đối tượng document truy cập phần tử → thiết lập thuộc tính cho phần tử.

```
function changeImage() {
  var image = document.getElementById("myImg");
  if (image.src.match("images/bulbon")) {
    image.src = "images/bulboff.gif";
    document.getElementById("btPower").innerHTML="Mở đèn";
  }
  else {
    image.src = "images/bulbon.gif";
    document.getElementById("btPower").innerHTML="Tắt đèn";
  }
}
```

112

112

Đối tượng document (tt)

- **Thay đổi style CSS của các phần tử:** Sử dụng đối tượng document truy cập phần tử → thiết lập style cho phần tử.

```
document.getElementById(id).style.property = new style;
hoặc
var obj = document.getElementById(id);
obj.style.property = new style;
```

113

113

Đối tượng document (tt)

- **Thêm phần tử vào DOM:**

- Sử dụng phương thức createElement() để tạo phần tử
- Thiết lập thuộc tính cho phần tử
- Thêm phần tử vào cây DOM bằng các phương thức appendChild(), insertBefore()

115

115

Đối tượng document (tt)

- **Thay đổi style CSS của các phần tử: ví dụ**

```
<body>
<p id="p1">Đoạn thứ nhất</p>
<p id="p2">Đoạn thứ hai được thay đổi style bằng JS</p>
<script>
document.getElementById("p2").style.color = "blue";
document.getElementById("p2").style.fontSize = "larger";
document.getElementById("p2").style.backgroundColor = "yellow";
</script>
</body>
```

114

114


Đối tượng document (tt)

- **Thêm phần tử vào DOM: ví dụ**

```
<div id="div1"> <p id="p1">Đoạn thứ 1.</p> <p id="p2">Đoạn thứ 2.</p> </div>
<script>
var p3 = document.createElement("p");
var text1 = document.createTextNode("Đây là đoạn mới");
p3.appendChild(text1);
var d = document.getElementById("div1");
d.appendChild(p3);
var h = document.createElement("h1");
var text2 = document.createTextNode("Đây là tiêu đề h1!");
h.appendChild(text2);
var p2 = document.getElementById("p2");
d.insertBefore(h, p2);
</script>
```

116

116


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Đối tượng document (tt)


- **Xóa phần tử trong DOM:**
 - Truy cập đến phần tử cha của phần tử cần xóa
 - Sử dụng phương thức `removeChild()`

```

<div id="div1">
  <p id="p1">1</p> <p id="p2">2</p> <p id="p3">3</p>
</div>
<script>
  var d = document.getElementById("div1");
  var p2 = document.getElementById("p2");
  d.removeChild(p2);
</script>

```


117


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Nội dung

- Giới thiệu JavaScript
- Ngôn ngữ kịch bản JavaScript
- Các đối tượng trong JavaScript
- **Xử lý sự kiện**
- Kiểm chứng dữ liệu
- Giới thiệu về đồ họa với JavaScript và HTML5

119


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Đối tượng document (tt)


- **Thay thế phần tử trong DOM:**
 - Truy cập đến phần tử cha của phần tử cần thay thế
 - Sử dụng phương thức `replaceChild()`

```

<div id="div1"> <p id="p1">1</p> <p id="p2">2</p> <p id="p3">3</p> </div>
<script>
  var d = document.getElementById("div1");
  var h = document.createElement("h1");
  var text2 = document.createTextNode("Đây là tiêu đề h1!");
  h.appendChild(text2);
  var p2 = document.getElementById("p2");
  d.replaceChild(h,p2);
</script>

```


118


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

4.4 Xử lý sự kiện

- Một số sự kiện có thể xảy ra trên trang web:
 - **onLoad**: xảy ra khi trang web, được nạp.
 - **onClick**: khi click chuột lên trên phần tử
 - **onDbIcClick**: khi nhấp đúp chuột lên trên phần tử.
 - **onMouseDown**: khi nhấn giữ chuột trên phần tử.
 - **onMouseUp**: nhả chuột sau khi nhấn chuột trên phần tử.
 - **onMouseOver, onMouseEnter**: con trỏ chuột đi vào phần tử
 - **onMouseMove**: khi di chuyển chuột trên phần tử.

120


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Xử lý sự kiện (tt)

- Một số sự kiện (tt):
 - onMouseOut, onMouseLeave**: khi con trỏ chuột rời khỏi phạm vi của phần tử.
 - onKeyDown**: khi nhấn phím trên phần tử đang nhận focus
 - onKeyUp**: khi nhả phím đã nhấn trước đó trên phần tử đang nhận focus
 - onKeyPress**: tương tự **onKeyDown**, nhưng không chứa thông tin các phím non-character như Shift, Ctrl, Esc, Tab, CapsLock, Alt, Backspace. Nếu cài đặt cả hai sự kiện thì **onKeyPress** được ưu tiên hơn
 - onChange**: Khi giá trị một phần tử đã thay đổi
 - onFocus**: khi phần tử nhận focus
 - onBlur**: khi phần tử mất focus
 - onSubmit**: khi form được submit
 - onReset**: khi form được reset

121


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Một số ví dụ về xử lý sự kiện

- Sự kiện **onClick**

```

<head>
  <script>
    function xemNgay() {
      document.getElementById("ngay").innerHTML = Date();
    }
  </script>
</head>
<body>
  <button id="btXem" onClick="xemNgay()">Xem ngày</button>
  <p id="ngay"></p>
</body>

```

123


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Một số ví dụ về xử lý sự kiện


- Sự kiện **onLoad**

```

<body onLoad="changeStyle()">
  <p id="demo"></p>
  <script>
    function changeStyle() {
      var p = document.getElementById("demo");
      p.style.color = "#ff0000";
      p.style.fontSize = "2em";
    }
  </script>
</body>

```

122


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Một số ví dụ về xử lý sự kiện


- Sự kiện **onChange**

```

<head>
  <script>
    function changeCase() {
      var x = document.getElementById("txtName");
      x.value = x.value.toUpperCase();
    }
  </script>
</head>
<body>
  Nhập vào tên bạn và nhấn tab hoặc enter:
  <input type="text" id="txtName" onchange="changeCase()">
</body>

```

124


 **Một số ví dụ về xử lý sự kiện**

- Sự kiện onFocus, onBlur

```
<script>
function setFocus(obj) {
    document.getElementById(obj.id).style.backgroundColor="#ffff00";
}
function setBlur(obj) {
    document.getElementById(obj.id).style.backgroundColor="#ffffff";
}
</script>

<input type="text" id="firstname" onFocus="setFocus(this)"
onBlur="setBlur(this)"> <br>
<input type="text" id="lastname" onFocus="setFocus(this)"
onBlur="setBlur(this)">
```


125

 **Một số ví dụ về xử lý sự kiện**

- Sự kiện onKeyDown, onKeyPress

```
<script>
function uniCharCode(event) {
    var char = event.which || event.keyCode;
    document.getElementById("demo").innerHTML = "The Unicode CHARACTER
code is: " + char;
}
function uniKeyCode(event) {
    var key = event.keyCode;
    document.getElementById("demo2").innerHTML = "The Unicode KEY code
is: " + key;
}
</script>
```

127


 **Một số ví dụ về xử lý sự kiện**

- Sự kiện onMouseOver, onMouseOut

```
<style> div{float:left width:50px; height:50px; margin:5px;} </style>
<script>
function bgChange(bg) {document.body.style.background = bg;}
</script>

<div style="background-color:red">
    onmouseover = "bgChange(this.style.backgroundColor)"
    onmouseout="bgChange('transparent') "
</div>
<div style="background-color:yellow">
    onmouseover="bgChange(this.style.backgroundColor)"
    onmouseout="bgChange('transparent') "
</div>
```

126

 **Một số ví dụ về xử lý sự kiện**


- Sự kiện onKeyDown, onKeyPress
 - onkeypress: thuộc tính keycode là mã Unicode của ký tự
 - onkeydown: thuộc tính keycode là mã ASCII của ký tự

onkeypress - The Unicode CHARACTER code is: 97

onkeydown - The Unicode KEY code is: 65

```
<input type="text" size="50"
onkeypress="uniCharCode(event)"
onkeydown="uniKeyCode(event)">
<p>onkeypress - <span id="demo"></span></p>
<p>onkeydown - <span id="demo2"></span></p>
```

128


 **Thêm trình xử lý sự kiện runtime**

- Có thể thêm một trình xử lý sự kiện cho phần tử lúc trang web đang thực thi:
 - Cách 1


```
document.getElementById(id).onclick=function() {
    //codes
}
```
 - Cách 2:


```
document.getElementById(id).addEventListener("click", functionname)
```

129

 **Nội dung**

- Giới thiệu JavaScript
- Ngôn ngữ kịch bản JavaScript
- Các đối tượng trong JavaScript
- Xử lý sự kiện
- **Kiểm chứng dữ liệu**
- Giới thiệu về đồ họa với JavaScript và HTML5


131

 **Thêm trình xử lý sự kiện runtime**

- Ví dụ:



```
<body>
<input type="button" id="btnClick" value="Click Me!!" />
<script>
    document.getElementById("btnClick").addEventListener("click", sayHello);
    function sayHello() {
        alert("Hello!");
    }
</script>
</body>
```

130

 **4.5 Kiểm chứng dữ liệu**

- Dữ liệu nhập từ người dùng được gửi về server để xử lý.
- Dữ liệu không hợp lệ được gửi về server gây hao tốn băng thông và server phải xử lý vô ích
- → Dữ liệu cần được kiểm tra tính hợp lệ trước khi gửi đi:
 - Bắt buộc nhập dữ liệu
 - Giới hạn giá trị số trong một khoảng cho trước
 - Kiểm chứng chuỗi nhập hợp lệ: số ký tự, địa chỉ email,...
- Kiểm chứng bằng mã JS hoặc tự động với HTML5


132

 **Kiểm chứng dữ liệu bằng mã Javascript**

- Viết hàm kiểm tra dữ liệu và gọi khi form được submit


```
<form name="fReg" action="" onSubmIt="return validation ()">
  <label for="txtUserName">User Name</label>
  <input type="text" id="txtUserName"><br>
  <label for="txtPassword">Password</label>
  <input type="password" id="txtPassword"> <br>
  <label for="txtConfirmPass">Confirm Password</label>
  <input type="password" id="txtConfirmPass"> <br>
  <input type="submit" value="Register">
</form>
<p id="info"></p>
```

133

 **Kiểm chứng dữ liệu tự động với HTML5**


- HTML5 cung cấp một số phần tử input mới có thể tự động kiểm chứng như:
 - <input type="url">
 - <input type="email">
 - <input type="number">
 - <input type="range">

135

 **Kiểm chứng dữ liệu bằng mã Javascript**


```
<script>
function validation(){
  let username = document.forms["fReg"]["txtUserName"].value;
  if(username == "") {
    document.getElementById("info").innerHTML="Please enter user name!";
    return false;
  }
  if(username.length > 20) {
    document.getElementById("info").innerHTML="user name must be <= 20 character!";
    return false;
  } //...
  return true;
}
</script>
```

134

 **Kiểm chứng dữ liệu tự động với HTML5**

- Các thuộc tính thường dùng:
 - required: bắt buộc nhập dữ liệu trong các phần tử input
 - max: giá trị lớn nhất (đối với số nguyên)
 - min: giá trị nhỏ nhất (đối với số nguyên)
 - minlength: số ký tự tối thiểu
 - maxlength: số ký tự tối đa
 - pattern: quy định chuỗi nhập phải thỏa một biểu thức chính quy
 - title: chuỗi gợi ý nhập liệu

136



TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Kiểm chứng dữ liệu (tt)

- Có thể kết hợp với CSS Pseudo Selectors

:disabled	Chọn các phần tử ở trạng thái bị vô hiệu hóa
:invalid	Chọn các phần tử input có dữ liệu nhập không hợp lệ
:optional	Chọn các phần tử input không sử dụng thuộc tính "required"
:required	Chọn các phần tử input có sử dụng thuộc tính "required"
:valid	Chọn các phần tử input có dữ liệu nhập hợp lệ


137


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Nội dung

- Giới thiệu JavaScript
- Ngôn ngữ kịch bản JavaScript
- Các đối tượng trong JavaScript
- Xử lý sự kiện
- Kiểm chứng dữ liệu
- Giới thiệu về đồ họa với JavaScript và HTML5**

139


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

Kiểm chứng dữ liệu (tt)

- Ví dụ


```

<style>
  :invalid {background-color: red;}
</style>

<form>
  <input type="text" name="UserName" required minlength="8"
  maxlength="20" pattern="[a-zA-Z0-9]+" title=" please enter 8 - 20
  characters " >
  <input type="number" name="YearOfBirth" min="1990" max="2000">
  <input type="submit" value="Đăng ký">
</form>

```

138


TRƯỜNG ĐẠI HỌC MỞ TP. HCM
 Cơ hội học tập cho mọi người

4.5. Đồ họa với Javascript và HTML5

- Sử dụng mã JS vẽ trên phần tử canvas.
- Phần tử canvas chứa đối tượng đồ họa:
 - đối tượng thiết bị ngữ cảnh
 - các công cụ để vẽ
- Mã lệnh JS thực hiện các thao tác vẽ và tô màu.

140

Đồ họa với Javascript và HTML5 (tt)

– Hệ thống tọa độ trong Canvas

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_shapes

141

Đồ họa với Javascript và HTML5 (tt)

• **Các thuộc tính về màu tô, nét vẽ:**

- fillStyle: màu tô
- strokeStyle: màu nét vẽ (color)
- shadowColor: màu của bóng (color)
- shadowBlur: độ mờ của bóng (số nguyên)
- shadowOffsetX: độ lệch của bóng theo chiều ngang (số nguyên)
- shadowOffsetY: độ lệch của bóng theo chiều dọc (số nguyên)
- lineWidth: độ dày nét vẽ (số nguyên)
- lineCap: đầu nét vẽ (butt|round|square)

143

Đồ họa với Javascript và HTML5 (tt)

• **Các bước thực hiện thao tác vẽ:**

- Tạo một biến tham chiếu đến phần tử canvas, sử dụng getElementById()
- Tạo đối tượng thiết bị ngữ cảnh từ canvas
- Thực hiện các thao tác vẽ và tô màu trên canvas
 - Bắt đầu vẽ: beginPath()
 - Sử dụng các phương thức: moveTo (), lineTo (), quadraticCurveTo (), bezierCurveTo (), arcTo () và arc ()
 - Sử dụng phương thức stroke() để thực sự vẽ

142

Đồ họa với Javascript và HTML5 (tt)

• **Các phương thức tạo màu tô:**

- createLinearGradient (x1, y1, x2, y2): tạo một màu tô pha trộn tuyến tính
 - x1, y1: vị trí điểm bắt đầu của gradient
 - x2, y2: vị trí điểm cuối của gradient
- createRadialGradient (x0, y0, r0, x1, y1, r1) tạo một đối tượng gradient tròn
 - x0, y0, r0: tọa độ tâm và bán kính hình tròn bắt đầu
 - x1, y1, r1: tọa độ tâm và bán kính hình tròn thứ hai
- addColorStop (stop, color): vị trí (0 - 1), màu sắc trong gradient, được sử dụng cùng với hai phương thức trên

144

Đồ họa với Javascript và HTML5 (tt)

- **Vẽ và tô màu hình chữ nhật:**
 - strokeRect(x, y, w, h): vẽ đường viền hình chữ nhật.
 - fillRect(x, y, w, h): tô màu vùng hình chữ nhật.
 - clearRect(x, y, w, h): xóa vùng hình chữ nhật.
 - Trong đó:
 - x, y: tọa độ góc trên trái HCN
 - w, h: chiều rộng, chiều cao HCN

145

Đồ họa với Javascript và HTML5 (tt)


- **Vẽ đoạn thẳng:** sử dụng các phương thức
 - moveTo(x1, y1): chuyển đến điểm (x1, y1)
 - lineTo(x2, y2): định vị trí điểm đích để vẽ từ điểm hiện hành đến điểm (x2, y2)
 - stroke(): thực hiện thao tác vẽ

147

Đồ họa với Javascript và HTML5 (tt)

- **Vẽ và tô màu hình chữ nhật (tt):**

```
<canvas id = "drawing" width = "200" height = "200"></canvas>
<script>
var canvas = document.getElementById("drawing");
var ctx = canvas.getContext('2d');
var grd = ctx.createLinearGradient(0, 0, 100, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "yellow");
ctx.fillStyle = grd;
ctx.fillRect(0, 0, 200, 200);
ctx.strokeStyle="rgb(0,0,255)";
ctx.strokeRect(0,0,200,200);
</script>
```




146

Đồ họa với Javascript và HTML5 (tt)

- **Vẽ đoạn thẳng (ví dụ):**

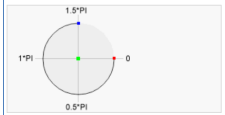
```
<canvas id = "drawing" width="200" height="100"> </canvas>
<script>
var c = document.getElementById("drawing");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.lineWidth = "5";
ctx.strokeStyle = "green";
ctx.moveTo(0, 75);
ctx.lineTo(250, 75);
ctx.stroke();
ctx.beginPath();
ctx.strokeStyle = "purple"; ctx.moveTo(50, 0);
ctx.lineTo(150, 130);
ctx.stroke();
</script>
```



148

Đồ họa với Javascript và HTML5 (tt)

- Vẽ đường tròn, cung tròn: Sử dụng các phương thức:
 - `arc(centerX, centerY, radius, startingAngle, endingAngle, counterclockwise)`: vẽ một cung tròn
 - Đường tròn là một cung khép kín, cần thiết lập:
 - `startingAngle = 0`
 - `endingAngle: 2*Math.PI`
 - `centerX, centerY`: tọa độ tâm đường tròn
 - `radius`: bán kính
 - `counterclockwise`:
 - `true`: ngược chiều kim đồng hồ
 - `false`: mặc định theo chiều kim đồng hồ



Center: `arc(100,75,50,0*Math.PI,1.5*Math.PI)`
 Start angle: `arc(100,75,50,0,1.5*Math.PI)`
 End angle: `arc(100,75,50,0*Math.PI,1.5*Math.PI)`

149

Đồ họa với Javascript và HTML5 (tt)

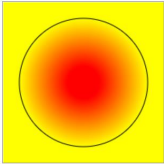
- Vẽ chữ**: Sử dụng các phương thức:
 - `strokeText(text, x, y)`: vẽ đường viền chữ
 - `fillText(text, x, y)`: tô màu chữ
 - `x, y` là vị trí điểm vẽ
 - Có thể thiết lập font chữ trước khi vẽ.

151

Đồ họa với Javascript và HTML5 (tt)

- Vẽ đường tròn (tt)**:


```
<canvas id="draw" width="200" height="200" style="border:1px solid gray;"> </canvas>
<script>
var canvas = document.getElementById("draw");
var ctx = canvas.getContext("2d");
ctx.beginPath();
var grd = ctx.createRadialGradient(100, 100, 20, 100, 100, 80);
grd.addColorStop(0, "red");
grd.addColorStop(1, "yellow");
ctx.fillStyle = grd;
ctx.fillRect(0, 0, 200, 200);
ctx.arc(100,100,80,0,2*Math.PI);
ctx.stroke();
</script>
```




150

Đồ họa với Javascript và HTML5 (tt)

- Vẽ chữ (tt)**:


```
<canvas id="myCanvas" width="500" height="200"> </canvas>
<script>
var canvas = document.getElementById('myCanvas');
ctx = canvas.getContext('2d');
ctx.strokeStyle = 'red';
ctx.fillStyle = 'yellow';
ctx.lineWidth = 2;
ctx.font = "bold 100pt Arial";
ctx.beginPath();
ctx.moveTo(50, 150); ctx.lineTo(450, 150);
ctx.stroke();
ctx.strokeStyle = 'blue';
ctx.fillText("Hello", 100, 150);
ctx.strokeText("Hello", 100, 150);
</script>
```



152

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Đồ họa với Javascript và HTML5 (tt)

- **Vẽ hình:**
 - Tạo đối tượng Image, thiết lập thuộc tính src cho đối tượng
 - Gọi phương thức drawImage
 - context.drawImage(img, x, y)
 - context.drawImage(img, x, y, width, height)

153

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Đồ họa với Javascript và HTML5 (tt)

- **Vẽ biểu đồ dùng thư viện Chart.js:**
 - Tham chiếu thư viện chart.js từ CDN


```
<script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.4/Chart.js"></script>
```
 - Khai báo phần tử <canvas> chứa biểu đồ


```
<canvas id="myChart" style="width:100%;max-width:700px"></canvas>
```
 - Vẽ biểu đồ theo cú pháp:


```
const myChart = new Chart("myChart", {
  type: "scatter",
  data: {},
  options: {}
});
```

Type:

 - Scatter Plot
 - Line Chart
 - Bar Chart
 - Pie Chart
 - Donut Chart
 - Bubble Chart
 - Area Chart
 - Radar Chart
 - Mixed Chart


155

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Đồ họa với Javascript và HTML5 (tt)

- **Vẽ hình (tt):**

```
<canvas id="myCanvas" width="300" height="200" ></canvas>
<script>
var canvas=document.getElementById("myCanvas");
var context=canvas.getContext("2d");
var imageObj = new Image();
imageObj.src = "images/monkey.png";
imageObj.onload = function() {
  context.drawImage(imageObj,0,0, 300, 200);
}
</script>
```



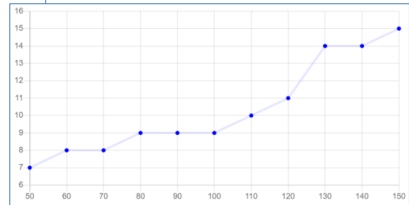
154

TRƯỜNG ĐẠI HỌC MỞ TP. HCM
Cơ hội học tập cho mọi người

Đồ họa với Javascript và HTML5 (tt)

- **Vẽ biểu đồ dùng thư viện Chart.js:**

```
<canvas id="myChart" style="width:100%;max-width:600px"></canvas>
<script>
const xValues = [50,60,70,80,90,100,110,120,130,140,150];
const yValues = [7,8,8,9,9,10,11,14,14,15];
new Chart("myChart", {
  type: "line",
  data: {
    labels: xValues,
    datasets: [{
      fill: false,
      lineTension: 0,
      backgroundColor: "rgba(0,0,255,1.0)",
      borderColor: "rgba(0,0,255,0.1)",
      data: yValues
    }]
  },
  options: {
    legend: {display: false},
    scales: {
      yAxes: [{ticks: {min: 6, max:16}}],
    }
  }
});
</script>
```



156



Đồ họa với Javascript và HTML5 (tt)

- **Vẽ biểu đồ dùng thư viện Google Chart:**
 - Tham chiếu thư viện



```
<script src="https://www.gstatic.com/charts/loader.js"></script>
```
 - Khai báo phần tử <div> chứa biểu đồ


```
<div id="myChart" style="max-width:700px; height:400px"></div>
```
 - Load the Google Graph API:
 - Load API và tcorechart package
 - Chỉ định gọi hàm callback


```
google.charts.load('current', {packages: ['corechart']});
google.charts.setOnLoadCallback(drawChart);
```

157

157




Đồ họa với Javascript và HTML5 (tt)

- **Vẽ biểu đồ dùng thư viện Google Chart:**

```
function drawChart() {
  var data = google.visualization.arrayToDataTable([
    ['Price', 'Size'],
    [50,7],[60,8],[70,8],[80,9],[90,9],
    [100,9],[110,10],[120,11],
    [130,14],[140,14],[150,15]
  ]);
  var options = {
    title: 'House Prices vs. Size',
    hAxis: {title: 'Square Meters'},
    vAxis: {title: 'Price in Millions'},
    legend: 'none'
  };
  var chart = new google.visualization.LineChart(document.getElementById('myChart'));
  chart.draw(data, options);
}
```

159

159



Đồ họa với Javascript và HTML5 (tt)

- **Vẽ biểu đồ dùng thư viện Google Chart:**
 - Tham chiếu thư viện


```
<script src="https://www.gstatic.com/charts/loader.js"></script>
```
 - Khai báo phần tử <div> chứa biểu đồ


```
<div id="myChart" style="max-width:700px; height:400px"></div>
```
 - Load the Google Graph API:
 - Load API và tcorechart package
 - Chỉ định gọi hàm callback


```
google.charts.load('current', {packages: ['corechart']});
google.charts.setOnLoadCallback(drawChart);
```

158

158