



1

Nội dung

- Giới thiệu về Javascript
- Nhúng Javascript vào trang web
- Cú pháp và kiểu dữ liệu Javascript
- Xử lý sự kiện
- DOM HTML với Javascript
- Bất đồng bộ
- Giới thiệu về ES6
- Giới thiệu về Biểu thức chính quy




 TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
UT-HCM

Phát triển ứng dụng Web

2

2



Giới thiệu Javascript




 TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
UT-HCM

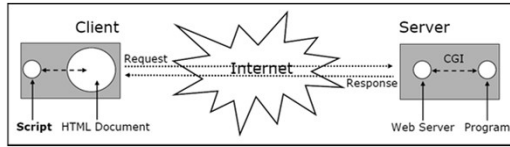
Phát triển ứng dụng Web

3

3

Ngôn ngữ kịch bản

- Kịch bản (Script) là những chương trình đơn giản hoạt động trên trình duyệt của người dùng (Client), hoặc ở phía server.



4

Ngôn ngữ kịch bản


- Ngôn ngữ kịch bản (Script language) là một ngôn ngữ lập trình, được dùng cho cả Client và Server
- Client Side Script:** thực thi tại Client-Side (trình duyệt):
 - Thực hiện các tương tác với người dùng (tạo menu chuyển động, ...), kiểm tra dữ liệu nhập, ...
 - JavaScript (Netscape, ban đầu chỉ hỗ trợ cho trình duyệt Netscape, nhưng nay đã hỗ trợ nhiều trình duyệt)
 - VBScript (Microsoft, hỗ trợ bởi IE)
- Server Side Script:** xử lý tại Server-Side
 - Nhằm tạo các trang web có khả năng phát sinh nội dung động.
 - Một số xử lý chính: kết nối CSDL, truy cập hệ thống file trên server, phát sinh nội dung html trả về người dùng...

5

Giới thiệu Javascript


- Là một ngôn ngữ kịch bản được phát triển Netscape Communications vào năm 1995, tên gọi ban đầu là LiveScript
- Sau này hợp tác với Sun Microsystems và đổi tên thành JavaScript.
- Chạy trên trình duyệt Netscape Navigation 2.0.
- Được hỗ trợ từ Internet Explorer 3.0.
- Có một số điểm chung với Java và sử dụng cú pháp tương tự ngôn ngữ C
- Được nhúng hoặc liên kết vào file HTML .
- Là ngôn ngữ Client-side script hoạt động trên trình duyệt của người dùng (client)
- Các ứng dụng client chạy trên trình duyệt như Netscape Navigator hoặc Internet Explorer, firefox, Chrome...

6




Giới thiệu Javascript

- JavaScript không phải là Java
- Là một ngôn ngữ lập trình đơn giản – ngôn ngữ kịch bản
- Chia sẻ xử lý trong ứng dụng web. Giảm các xử lý không cần thiết trên server.
- Điều khiển tính năng của trình duyệt
 - Hiện một cửa sổ hoặc một thông điệp đơn giản
- Giúp tạo các hiệu ứng, tương tác cho trang web
- Lưu trữ và sử dụng thông tin của người dùng
 - Tạo và đọc Cookie
- JavaScript làm cho việc tạo các trang Web động và tương tác dễ dàng hơn
 - Cung cấp sự tương tác cho người dùng
 - Thay đổi nội dung động
 - Xác nhận tính hợp lệ của dữ liệu


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN


Phát triển ứng dụng Web 7

7




Java và Javascript

- Java và Javascript khác nhau thế nào?
- Tại sao lại có "Java" trong tên Javascript?



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 8


8



Java và Javascript



Java	JavaScript
Java là ngôn ngữ lập trình tĩnh, hướng đối tượng, hoạt động trên nhiều nền tảng.	JavaScript là ngôn ngữ lập trình động (hay ngôn ngữ kịch bản - scripted language) được sử dụng để làm cho các trang web trở nên sinh động
Java là ngôn ngữ được biên dịch (compiled)	JavaScript là ngôn ngữ được diễn giải/thông dịch (interpreted).
Java là một ngôn ngữ độc lập	JavaScript phụ thuộc nhiều hơn, nghĩa là nó hoạt động với HTML và CSS trên các trang web để tạo nội dung động.


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

9

Giới thiệu Javascript

- Vào thời điểm JavaScript được tạo ra (năm 1995), Java là một ngôn ngữ lập trình mới, rất phổ biến và được kỳ vọng sẽ thay đổi ngành công nghiệp phần mềm. Netscape (công ty tạo ra JavaScript) muốn tận dụng sự nổi tiếng của Java để quảng bá cho ngôn ngữ kịch bản mới của họ.
- Ban đầu, JavaScript có tên là LiveScript, nhưng sau đó được đổi tên thành JavaScript để thu hút sự chú ý của các lập trình viên Java và cộng đồng phát triển web.
- Mặc dù có tên gọi tương tự, Java và JavaScript là hai ngôn ngữ hoàn toàn khác biệt. Việc sử dụng 'Java' trong tên JavaScript chỉ là chiến lược marketing vào thời điểm đó

10

Nhúng Javascript vào HTML

- Sử dụng thẻ SCRIPT:


```
<script>
    JavaScript statements;
</script>
```
- Sử dụng một file JavaScript ở ngoài


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

11

Nhúng Javascript vào HTML

```
<html>
<head>
  <script>
    some statements
  </script>
</head>
<body>
  <script>
    some statements
  </script>
  <script src="Tên_file_script.js"></script>
  <script>
    //gọi thực hiện các phương thức được định nghĩa
    //trong "Tên_file_script.js"
  </script>
</body>
</html>
```

12

Nhúng Javascript vào HTML

- Đặt giữa tag <head> và </head>: script sẽ **thực thi** ngay khi trang **web được mở**, không sử dụng async hoặc defer.
- Đặt giữa tag <body> và </body>: script trong phần body được **thực thi sau** khi thực thi các **đoạn script** có trong phần <head>.
- Số lượng đoạn client-script chèn vào trang **không hạn chế**.

13

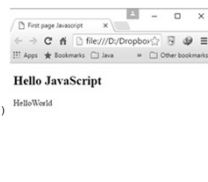
Nhúng Javascript vào HTML

- Thuộc tính của thẻ <script>:
 - type: Xác định kiểu script, thường là "text/javascript" (có thể bỏ qua).
 - src: Đường dẫn đến file Javascript bên ngoài.
 - async: Tải script bất đồng bộ, không chặn render trang web.
 - defer: Tải script bất đồng bộ, thực thi sau khi trang web đã được tải xong.
- Ví dụ:
 - <script src="myscript.js"></script>
 - <script async src="myscript.js"></script>
 - <script defer src="myscript.js"></script>

14

Ví dụ


```
<!DOCTYPE html>
<html>
<head>
  <title>First page Javascript</title>
  <script>
    document.writeln("<H2>Hello JavaScript</H2>")
    document.write("HelloWorld");
  </script>
</head>
<body>
</body>
</html>
```



15




Cú pháp và kiểu dữ liệu trong Javascript


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN


Phát triển ứng dụng Web 16

16




Cú pháp

- Phân biệt chữ hoa và thường
- Khoảng trắng, tab, xuống dòng chỉ dùng trong chuỗi.
- Kết thúc câu lệnh là dấu chấm phẩy ";"
- Dấu phẩy để phân biệt các phần tử trong mảng


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN


Phát triển ứng dụng Web 17

17



Các quy tắc chung

- Khởi lệnh được bao trong dấu {}
- Mỗi lệnh nên kết thúc bằng dấu ;
- Cách ghi chú thích:
 - // Chú thích 1 dòng
 - /* Chú thích nhiều dòng */


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 18

18

Biến



- **Phân biệt** chữ hoa và thường.
 - Ví dụ : Hai biến Java, java là khác nhau
- **Cách đặt tên biến**
 - Tuân thủ theo nguyên tắc đặt tên như lập trình C
 - A..Z,a..z,0..9_ : phân biệt HOA, Thường
- **Khai báo biến**
 - Sử dụng từ khóa var
 - Ví dụ: **var count=10, amount;**
 - Không cần khai báo biến trước khi sử dụng, biến thật sự tồn tại khi bắt đầu sử dụng lần đầu tiên



Phát triển ứng dụng Web 19

19

[illegible]

Biến



- **Đặc điểm của cách khai báo var.**
 - ❑ Biến var có tính chất hoisting: nghĩa là dù khai báo ở đâu trong hàm thì biến này đều sẽ được đem lên đầu scope trước khi code được thực hiện, nhưng giá trị khởi tạo không được nâng lên.
 - ❑ Ví dụ:

```
console.log(greeting);  
var greeting = "Chào mấy nì";
```
 - ❑ Sẽ được biên dịch là:

```
var greeting;  
console.log(greeting); // greeting is undefined  
greeting = "Chào mấy nì";
```



Phát triển ứng dụng Web 20

20

Biến



- **Đặc điểm của cách khai báo var.**
 - ❑ Vấn đề:

```
var greeting = "Chào mấy ní ";
var times = 4;
if (times > 3) {
  var greeting = " Bảnh chào mọi người nhé ! ";
}
```

`console.log(greeting);`
 - ❑ Vấn đề: `greeting` khi này sẽ có giá trị là " Bảnh chào mọi người nhé ! " và sẽ vẫn giữ nguyên giá trị này sau khi thoát ra khỏi block if



Phát triển ứng dụng Web 21

21

Biến



- Trong ES 6 (viết tắt của ECMAScript 6) đã bổ sung **let** và **const** phục vụ cho việc khai báo biến dữ liệu.
- **let**: Khai báo biến với phạm vi khối lệnh (block scope). Tránh lỗi khi biến bị ghi đè hoặc sử dụng ngoài phạm vi.
- **const**: Khai báo hằng số, giá trị không thể thay đổi sau khi gán. Sử dụng cho các giá trị không thay đổi trong suốt chương trình.
- Biến let và const chỉ tồn tại trong block code nơi chúng được khai báo.

22

Biến



```
// let có thể thay đổi giá trị
let count = 1;
count = 2; // Hợp lệ
if (true) {
  let x = 10; // Biến x chỉ tồn tại trong khối lệnh này
  console.log(x); // 10
}
console.log(x); // Lỗi: x không được định nghĩa
const PI = 3.14; // Hằng số PI
PI = 3.15; // Lỗi: Không thể thay đổi giá trị của PI
```

23

Biến



```
Ví dụ:
const person = { name: "John", age: 30 };

// Hợp lệ: Thay đổi thuộc tính của object
person.age = 31;
console.log(person.age); // 31

// Lỗi: Gán lại biến 'const'
// person = { name: "Alice" };

console.log(person); // { name: "John", age: 31 }
```

24

Biến

- Không trùng với từ khoá của JavaScript

abstract delete innerWith Packages status
 alert do instanceof pageOffset statusbar
 arguments document int pageOffset statusbar
 Array double interface parent String
 blur else isFinite parentFunction super
 boolean escape java parentElement switch
 Boolean escape java parentElement switch
 break eval length parseInt parseInt
 byte export location private this
 caller extends locationbar prompt throw
 catch finally link prototype top
 case find member public toString
 catch host moveBy Rollup transparent
 char focus moveTo releaseEvents try
 class for name resizeBy unescape
 clearInterval frames NaN resultTo unescape
 clearTimeout Function native return unescape
 close function netscape valueOf var
 closed goto new scroll void
 confirm history null scrollbar watch
 console home Number scrollBy while
 constructor if Object scrollTo white
 continue implements open self window
 Date import opener setAttribute window
 debugger is outerHeight setTimeout with
 default Infinity outerHTML static FALSE
 defaultStatus innerHeight package true

Phát triển ứng dụng Web 25

25

Kiểu dữ liệu

Kiểu dữ liệu	Ví dụ	Mô tả
Object	<code>var listBooks = new Array(10);</code>	Trước khi sử dụng, phải cấp phát bằng từ khóa new
String	"The cow jumped over the moon." "40"	Chứa được chuỗi unicode Chuỗi rỗng ""
Number	0.066218 12	Theo chuẩn IEEE 754
boolean	true / false	
undefined	<code>var myVariable;</code>	<code>myVariable = undefined</code>
null	<code>connection.Close();</code>	<code>connection = null</code>

Phát triển ứng dụng Web 26

26

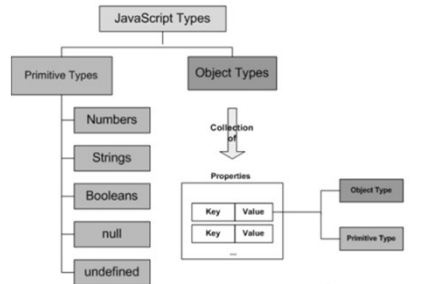
Kiểu dữ liệu

- Một biến trong Javascript có thể lưu bất kỳ kiểu dữ liệu nào
- typeof operator: Kiểm tra kiểu dữ liệu của biến.
 - `console.log(typeof 10); // "number"`
 - `console.log(typeof "hello"); // "string"`
- Trong JavaScript, hai biến khác kiểu có thể kết hợp với nhau.
 - Ví dụ: `A = "This apple costs Rs." + 5`
Kết quả: một chuỗi với giá trị là "This apple costs Rs. 5".

Phát triển ứng dụng Web 27

27

Kiểu dữ liệu



Phát triển ứng dụng Web 28

28

Dynamic Typing và Type Coercion

- Dynamic Typing: Kiểu dữ liệu của biến được xác định trong quá trình chạy, có thể thay đổi.
- Type Coercion: Javascript tự động chuyển đổi kiểu dữ liệu khi thực hiện phép toán giữa các biến khác kiểu.
- `let x = 10;` // x là kiểu Number
- `x = "Hello";` // x bây giờ là kiểu String
- `console.log(1 + "2");` // "12" (chuyển đổi Number thành String)

Phát triển ứng dụng Web 29

29

Null, Undefined và NaN

Phát triển ứng dụng Web 30

30

Null, Undefined và NaN

- Null là đại diện cho một giá trị không tồn tại (giá trị "không có"). Nó cần phải được gán cho một biến khi muốn biến không chứa giá trị

```
var a = null;
console.log(a); //null
```

31

Null, Undefined và NaN

- undefined có nghĩa là không xác định
 - 1 biến được khai báo nhưng không được gán giá trị
 - Truy cập giá trị của một thuộc tính không tồn tại
 - Giá trị trả về của 1 hàm không trả về 1 giá trị:
 - Giá trị tham số của hàm đã được khai báo nhưng bị bỏ qua tham số khi gọi hàm

32

Null, Undefined và NaN

- Sự khác biệt giữa null và undefined
 - undefined nghĩa là một biến đã được khai báo nhưng chưa được gán giá trị, null là được gán cho một biến.

33

Null, Undefined và NaN



- NaN là viết tắt của "Not a Number". Khi một function hoặc operation trong JavaScript không thể trả về một số cụ thể, nó sẽ trả về giá trị NaN thay thế. NaN sẽ được coi như một kiểu Number
- Một số trường hợp sinh ra NaN:
 - Lấy số 0 chia cho số 0
 - Lấy vô cùng (infinity) chia cho vô cùng (infinity)
 - Nhân vô cùng (infinity) với số 0
 - Bất kỳ phép tính toán nào trong đó NaN là một toán hạng
 - Chuyển đổi một xâu non-numeric hoặc undefined về dạng number.

34

Null, Undefined và NaN



- `let x;` // x là undefined
- `let y = null;` // y là null
- `console.log(0 / 0);` // NaN

35

Đổi kiểu dữ liệu



- Biến tự đổi kiểu dữ liệu khi giá trị mà nó lưu trữ thay đổi
 - Ví dụ:


```
var x = 10;           // x kiểu Number
x = "hello world !";  // x kiểu String
```
- Có thể cộng 2 biến khác kiểu dữ liệu
 - Ví dụ:


```
var x;
x = "12" + 34.5;      // KQ: x = "1234.5"
```
- Hàm **parseInt(...)**, **parseFloat(...)**: Đổi kiểu dữ liệu từ chuỗi sang số.

36

Kiểu chuỗi

- Một hằng chuỗi được nằm giữa hai dấu nháy đôi ""
 - Ví dụ: `var chuoi = "Đây là Kiểu Chuỗi";`
- Các phép toán trên chuỗi
 - Phép nối chuỗi: `+, +=`
 - Phép so sánh: `<, <=, >, >=, ==, !=`
 - Phương thức xử lý: `indexOf()`, `lastIndexOf()`, `substring()`, `split()`, `replace()`, `toUpperCase()`, `toLowerCase()`,...

```
let str = "Hello, World!";
console.log(str.length); // 13 (độ dài chuỗi)
console.log(str.charAt(0)); // "H" (ký tự tại vị trí đầu tiên)
console.log(str.substring(0, 5)); // "Hello" (trích xuất chuỗi)
```



Trường Đại học Công nghệ Thông tin
UIT-HCM

40

40

Kiểu luận lý

- Một biến có kiểu luận lý tồn tại 1 trong 2 trạng thái : **true, false**.
 - Ví dụ: `var t = true, f = false;`
- Các phép toán trên kiểu luận lý
 - Phép so sánh : `<, <=, >, >=, ==, !=`
 - Phép logic : `&&` (và), `||` (hoặc), `!` (phủ định).



Trường Đại học Công nghệ Thông tin
UIT-HCM

Phát triển ứng dụng Web

41

41

Kiểu luận lý – Ví dụ

```
<script>
var x = 10;
var y = 5;
document.write("The value of x is " + x + "
The value of y is " + y + "<br>");
document.write("x AND y = " + (x &&
y) + "<br>");
document.write("x OR y = " + (x || y) + "<br>");
document.write("NOT x = " + (!x) + "<br>");
</script>
```



Trường Đại học Công nghệ Thông tin
UIT-HCM

Phát triển ứng dụng Web

42

42

Mảng

- **Mảng một chiều:**

- o var A = new Array(10)
- o Mảng A nói trên có 10 phần tử, và chỉ số phần tử đầu tiên của mảng bắt đầu 0, muốn truy xuất đến phần tử có chỉ số i, ta dùng **A[i]**.

// Cách khai báo mảng chuẩn

```
let arr = [1, 2, 3, 4, 5];
```

- Mảng hai chiều

- Khai báo A là mảng 2 chiều có 10 dòng, 20 cột.

```
var A = new Array(10), i = 0;
for (i = 0; i < 10; i++)
    A[i] = new Array(20);
```
- Để truy xuất đến phần tử có chỉ số dòng i, chỉ số cột j ta dùng A[i][j].



Phát triển ứng dụng Web 43

43

Mảng

- Phương thức thao tác với mảng:

- `push()`, `pop()`: Thêm/xóa phần tử cuối mảng.
- `shift()`, `unshift()`: Thêm/xóa phần tử đầu mảng.
- `slice()`: Tạo mảng con.
- `splice()`: Thêm/xóa/thay thế phần tử trong mảng.
- `concat()`: Nối hai mảng.
- `forEach()`: Duyệt qua từng phần tử mảng.
- `map()`: Tạo mảng mới bằng cách biến đổi từng phần tử mảng.
- `filter()`: Tạo mảng mới chứa các phần tử thỏa mãn điều kiện.
- `reduce()`: Gom các phần tử mảng thành một giá trị duy nhất.



Phát triển ứng dụng Web 44

44

Lệnh rẽ nhánh

- Câu lệnh điều kiện được dùng để kiểm tra điều kiện. Kết quả xác định câu lệnh hoặc khối lệnh được thực thi.

- Các câu lệnh điều kiện bao gồm:

- **if**
- **if..... else**
- **Switch**



Phát triển ứng dụng Web 45

45

Câu lệnh if

- Dùng để xử lý lệnh khi biểu thức của if trả về giá trị true

```
if (biểu thức điều kiện)
    Khối lệnh;
```

- Ví dụ:

```
a= "mon";  
if (a=="tue")  
    document.write("Hôm nay được nghỉ");
```



Phát triển ứng dụng Web 46

46

Câu lệnh if...else

```
if (biểu thức điều kiện)
    Khối lệnh 1;
```

```
else
    Khối lệnh 2;
```

- Ví dụ:

```
a=5;
if (a%2==0)
    document.write(a, "là số chẵn");
else
    document.write(a, "là số lẻ");
```



Phát triển ứng dụng Web 47

47

Câu lệnh switch...case

switch (biến hoặc biểu thức)

```
{
    case giá trị 1:
        Khối lệnh 1;
        break;
    case giá trị 2:
        Khối lệnh 2;
        break;
    ...
    default:
        Khối lệnh n;
}
```



Phát triển ứng dụng Web 48

48

Câu lệnh switch...case – Ví dụ

```
var diem = "G";
switch (diem) {
  case "Y":
    document.write("Yếu");
    break;
  case "TB":
    document.write("Trung bình");
    break;
  case "K":
    document.write("Khá");
    break;
  case "G":
    document.write("Giỏi");
    break;
  default:
    document.write("Xuất sắc")
}
```

Trường Đại Học Công Nghệ Hoàng Quốc Việt

Phát triển ứng dụng Web 49

49

Lệnh lặp

- Cấu trúc điều khiển lặp trong chương trình là các lệnh lặp
- Các kiểu lệnh lặp bao gồm:
 - **for**
 - **do While**
 - **while**

Trường Đại Học Công Nghệ Hoàng Quốc Việt

Phát triển ứng dụng Web 50

50

Lệnh lặp for

```
for (biểu thức1; biểu thức 2; biểu thức3){
  Khối lệnh;
}
```

- Biểu thức 1: khởi tạo giá trị
- Biểu thức 2: điều kiện
- Biểu thức 3: cập nhật giá trị

- Khối lệnh được thực hiện khi biểu thức 2 còn đúng.
- Ví dụ: **for** (i = 0; i < 10; i++)
s+=2*i;

Trường Đại Học Công Nghệ Hoàng Quốc Việt

Phát triển ứng dụng Web 51

51

Lệnh lặp while

```
while (biểu thức điều kiện) {  
    Khối lệnh;  
}
```

- Khối lệnh được thực hiện khi biểu thức trong while còn đúng.

- Ví dụ:

```
i=0;  
while (i<20) {  
    s+=i;  
    i++;  
}
```

52

Hàm

- Dùng từ khóa **function** để khai báo hàm. Muốn trả về giá trị của hàm ta dùng từ khóa **return**.

- Dạng thức khai báo chung:

```
function Tên_hàm(thamso1, thamso2,..)  
{  
    Khối lệnh;  
}
```

- Hàm có giá trị trả về:

```
function Tên_hàm(thamso1, thamso2,..)  
{  
    Khối lệnh;  
    return (value);  
}
```

53

Ví dụ

```
function Add(x,y)  
{  
    return (x+y);  
}  
  
var t;  
t = Add(4,8);  
document.write(t);
```

54

Hàm eval

- Biến chuỗi thành biểu thức
- Biến chuỗi thành lệnh
- Biểu thức có thể bao gồm nhiều biến và nhiều thuộc tính của một đối tượng.

```
var x = 5;
var y = 10;
document.write(eval("x+y+5"));
```



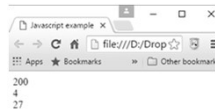
Phát triển ứng dụng Web

55

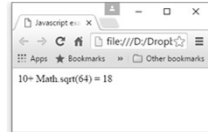
55

Ví dụ hàm eval

```
<script>
  eval("x=10;y=20;document.write(x*y)");
  document.write("<br>" + eval("2+2"));
  document.write("<br>" + eval(x+17));
</script>
```



```
<script>
  var string= "10+ Math.sqrt(64)";
  document.write(string + "=" + eval(string));
</script>
```



Phát triển ứng dụng Web

56

56

Ví dụ hàm eval

- Thực thi mã JavaScript dưới dạng một chuỗi ký tự (string). Hàm này có thể đánh giá (evaluate) các biểu thức hoặc chạy các đoạn mã mà dev truyền vào nên eval() không an toàn vì có thể thực thi bất kỳ mã nào, dễ dẫn đến các lỗ hổng bảo mật. Ví dụ: XSS (Cross-site Scripting)

- Thay thế: (Kiểm tra và xác thực đầu vào trước):

// Thay vì sử dụng eval để tính toán biểu thức, hãy dùng Function

```
let expression = "5 + 5";
```

```
let result = Function('use strict'; return (${expression}))();
```

```
console.log(result); // 10
```



Phát triển ứng dụng Web

57

57

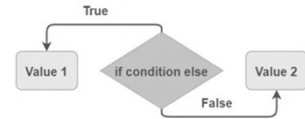
Toán tử ba ngôi

Conditional (ternary) operator

Cú pháp: **(Điều kiện) ? value1: value2**

Nếu biểu thức điều kiện đúng thì trả về giá trị value 1

Nếu biểu thức điều kiện sai thì trả về giá trị value 2



```

<script>
  var exp = 1;
  var salary = exp > 3 ? 1000 : 500;
  console.log(salary) //500
</script>
  
```

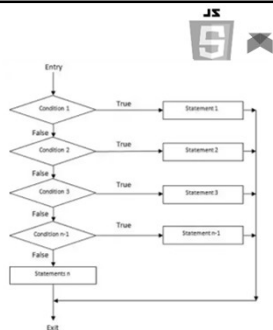
58

Toán tử ba ngôi

Nó có thể được gọi “nối tiếp” theo cách sau, tương tự như với If - else If - else If - else nối tiếp nhau

```

<script>
  var exp = 2;
  var salary = exp < 1 ? 1000 :
    exp < 2 ? 1500 :
    exp < 3 ? 2000: 3000;
  console.log(salary) //2000
</script>
  
```



59

Toán tử ba ngôi

```

let age = 18;
let canVote = age >= 18 ? "Được bỏ phiếu" : "Không được bỏ phiếu";
console.log(canVote); // "Được bỏ phiếu"
// Sử dụng toán tử ba ngôi lồng nhau
let status = age < 13 ? "Trẻ em" : age < 18 ? "Thiếu niên" : "Người lớn";
console.log(status); // "Người lớn"
  
```

60

Nullish Coalescing Operator (??)

Toán tử điều kiện

Nullish chính là NULL or UNDEFINED

Cú pháp: **a ?? b**

Nếu a được định nghĩa thì kết quả sẽ là a

Nếu a là Nullish thì kết quả sẽ là b

```
let a = NULL
console.log(a??50) //50
console.log(a) //NULL
let a=10
let c=30
console.log(a??b??c??d) //10
```



Phát triển ứng dụng Web 61

61

Logical Nullish Assignment (??=)

Cú pháp: **a ??= b** tương đương **a ??(a=b)**

Nếu a không phải là NULLISH (NULL or UNDEFINED) thì kết quả sẽ là a.

Nếu a là NULLISH thì kết quả sẽ là b, nhưng b sẽ có nhiệm vụ gán giá trị cho a. Hay gọi là assigned.

```
let a = NULL
console.log(a??=50) //50
console.log(a) //50
```



Phát triển ứng dụng Web 62

62

Các đối tượng cơ bản trong Javascript

- Lớp đối tượng là tập các đối tượng có cùng thuộc tính và hành vi (phương thức)
- Thuộc tính (biến): dùng để định nghĩa đối tượng.
- Phương thức (hàm): những hoạt động của lớp đối tượng
- Ví dụ:

```
let person = {
  name: "John",
  age: 30,
  greet: function() {
    console.log("Hello, " + this.name);
  }
};
person.greet(); // "Hello, John"
```



Phát triển ứng dụng Web 63

63

Các đối tượng cơ bản trong Javascript

• Ví dụ lớp:

```
class Animal {
  constructor(name, species) {
    this.name = name;
    this.species = species;
  }
  makeSound() {
    console.log(`${this.name} makes a sound.`);
  }
}
const dog = new Animal("Buddy", "Dog");
dog.makeSound(); // "Buddy makes a sound."
```

64

Các đối tượng cơ bản trong Javascript

- JavaScript hỗ trợ OOP, nhưng theo cách prototype-based (dựa trên nguyên mẫu), khác với class-based (dựa trên lớp) như Java, C++.
- Nguyên mẫu (Prototype): Mỗi object trong Javascript có một prototype (nguyên mẫu), là một object khác mà object hiện tại có thể kế thừa thuộc tính và phương thức.
- Kế thừa (Inheritance): Object có thể kế thừa thuộc tính và phương thức từ prototype của nó.
- Tính đóng gói (Encapsulation): Che giấu dữ liệu bên trong object, chỉ cho phép truy cập và thay đổi thông qua các phương thức.
- Tính đa hình (Polymorphism): Cho phép các object khác nhau phản hồi cùng một phương thức theo cách riêng.

65

Các đối tượng cơ bản trong Javascript

- Khác với class-based programming (lập trình hướng lớp) sử dụng các lớp (classes) làm bản thiết kế để tạo object, prototype-based programming sử dụng object hiện có làm nguyên mẫu để tạo object mới.
- Mỗi object trong JavaScript đều có một thuộc tính ẩn gọi là `__proto__`, trỏ đến prototype (nguyên mẫu) của nó.
- Prototype cũng là một object, và nó cũng có thể có prototype riêng của mình, tạo thành một chuỗi prototype (prototype chain).
- Khi truy cập một thuộc tính hoặc phương thức của một object, Javascript sẽ tìm kiếm thuộc tính/phương thức đó trong object hiện tại.
- Nếu không tìm thấy, nó sẽ tiếp tục tìm kiếm trong prototype của object.
- Quá trình này tiếp tục cho đến khi tìm thấy thuộc tính/phương thức hoặc đến cuối chuỗi prototype (null).

66

Các đối tượng cơ bản trong Javascript

```
// Hàm constructor
function Animal(name) {
  this.name = name;
}
// Thêm phương thức makeSound() vào prototype của Animal
Animal.prototype.makeSound = function() {
  console.log(`${this.name} makes a sound.`);
};
// Tạo object mới từ hàm constructor
const cat = new Animal("Cat");
// Gọi phương thức makeSound()
cat.makeSound(); // "Cat makes a sound."
// Kiểm tra prototype của cat
console.log(cat.__proto__ === Animal.prototype); // true
```

Phát triển ứng dụng Web

67

67

Các đối tượng cơ bản trong Javascript

- Ưu điểm của Prototype-based Programming:
 - Linh hoạt: Dễ dàng thêm thuộc tính và phương thức cho object, prototype trong quá trình chạy.
 - Hiệu quả: Tái sử dụng code thông qua kế thừa prototype, tránh lặp code.
 - Đơn giản: Cú pháp đơn giản hơn class-based programming.
- Nhược điểm:
 - Khó hiểu: Khái niệm prototype chain có thể khó hiểu đối với người mới học.
 - Khó debug: Lỗi trong chuỗi prototype có thể khó debug.

Phát triển ứng dụng Web

68

68

Thuộc tính và phương thức

- Để truy cập thuộc tính của đối tượng, chúng ta phải chỉ ra tên đối tượng và thuộc tính của nó:


```
objectName.propertyName
```
- Để truy cập phương thức của đối tượng, chúng ta phải chỉ ra tên đối tượng và thuộc tính của nó:


```
objectName.method()
```
- Tìm hiểu về Optional Chaining Operator (?.)
 - Cú pháp: `obj ?. prop`

Phát triển ứng dụng Web

69

69

Thuộc tính và phương thức

- Optional Chaining Operator (?) là một toán tử được giới thiệu trong ECMAScript 2020 (ES11), cho phép truy cập thuộc tính hoặc gọi phương thức của một object một cách an toàn, ngay cả khi object hoặc các thuộc tính là null hoặc undefined.
 - Optional Chaining Operator (?) giúp rút ngắn và đơn giản hóa code bằng cách ngắn mạch (short-circuit) quá trình truy cập nếu gặp null hoặc undefined.
- ```
const street = user?.address?.street;
```
- `user?.address` : Nếu `user` là null hoặc undefined, biểu thức sẽ trả về undefined ngay lập tức, không thực hiện truy cập `address`.
- `user?.address?.street`: Nếu `user` hoặc `address` là null hoặc undefined, biểu thức sẽ trả về undefined ngay lập tức.

70

---

---

---

---

---

---

---

---

## Thuộc tính và phương thức

```
const user = {
 name: "John",
 address: {
 street: "123 Main St",
 },
};

// Kiểm tra thủ công
const street = user && user.address && user.address.street;
console.log(street); // "123 Main St"

// Sử dụng Optional Chaining
const streetNew = user?.address?.street;
console.log(streetNew); // "123 Main St"
```

71

---

---

---

---

---

---

---

---

## Sử dụng đối tượng

- Khi tạo trang web, chúng ta cần sử dụng:
  - Các đối tượng trình duyệt
  - Các đối tượng có sẵn (thay đổi phụ thuộc vào ngôn ngữ kịch bản được sử dụng)
  - HTML elements
- Chúng ta cũng có thể tạo ra các đối tượng để sử dụng theo yêu cầu của mình.
  - Object literal: Cách tạo đối tượng bằng {}.
  - Constructor function: Cách tạo đối tượng bằng hàm constructor.
  - Class: Cách tạo đối tượng bằng cú pháp class (ES6).

72

---

---

---

---

---

---

---

---



## Từ khóa this

- Giá trị của nó chỉ ra đối tượng hiện hành và có thể có các thuộc tính chuẩn, chẳng hạn như tên, độ dài, và giá trị được áp dụng phù hợp

```
<script>
function person(first_name, last_name, age, sex){
 this.first_name=first_name;
 this.last_name=last_name;
 this.age=age;
 this.sex=sex;
 this.printStats=printStats;
}
function printStats(){
 document.write("Name : " + this.last_name + " " + this.first_name + "
");
 document.write("Age : "+this.age+"
");
 document.write("Sex : "+this.sex+"
");
}
</script>
```



Phát triển ứng dụng Web 73

73

---

---

---

---

---

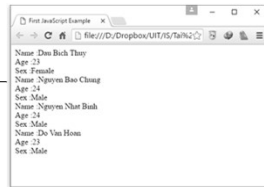
---

---

---

## Từ khóa this

```
person1= new person("Thuy", "Dau Bich", "23", "Female");
person2= new person("Chung", "Nguyen Bao", "24", "Male");
person3= new person("Binh", "Nguyen Nhat", "24", "Male");
person4= new person("Hoan", "Do Van", "23", "Male");
person1.printStats();
person2.printStats();
person3.printStats();
person4.printStats();
</script>
```



Phát triển ứng dụng Web 74

74

---

---

---

---

---

---

---

---

## Từ khóa this

- Các trường hợp sử dụng this:
  - Trong method của object.
  - Trong hàm constructor.
  - Trong hàm thông thường (không ràng buộc với object.)



Phát triển ứng dụng Web 75

75

---

---

---

---

---

---

---

---

## Từ khóa this

- Trong method của object.

```
const person = {
 name: "John",
 greet: function() {
 console.log('Hello, my name is ${this.name}'); // 'this' tham chiếu đến object 'person'
 }
};
person.greet(); // "Hello, my name is John"
```

76

---

---

---

---

---

---

---

---

## Từ khóa this

- Trong hàm constructor.

```
function Car(brand) {
 this.brand = brand; // 'this' tham chiếu đến object 'Car' được tạo ra
}
const myCar = new Car("Ford");
console.log(myCar.brand); // "Ford"
```

77

---

---

---

---

---

---

---

---

## Từ khóa this

- Trong hàm thông thường (không ràng buộc với object).

```
function myFunction() {
 console.log(this); // 'this' tham chiếu đến global object (window trong browser)
}
myFunction(); // window
```

78

---

---

---

---

---

---

---

---

## Lệnh for...in

- Câu lệnh **for...in** được dùng để lặp mỗi thuộc tính của đối tượng hoặc mỗi phần tử của một mảng.

- Cú pháp:

```
for (variable in object) {
 statements;
}
```

- Lưu ý rằng for...in thường được sử dụng để duyệt qua thuộc tính của object, không nên sử dụng để duyệt qua mảng vì thứ tự không được đảm bảo.

79

---

---

---

---

---

---

---

---

## Lệnh for...in

```
<script>
function myFunction() {
 var x;
 var txt="";
 var person={Ho:"Nguyen Van",Ten:"Teo",age:27};
 for (x in person) {
 txt=txt + person[x];
 }
 document.getElementById("demo").innerHTML=txt;
}
</script>
</head>
<body>
<p>Nhấn nút để hiển thị tên</p>
<button onclick="myFunction()">Lấy thông tin</button>
<p id="demo">đây là hiển thị.</p>
</body>
```

80

---

---

---

---

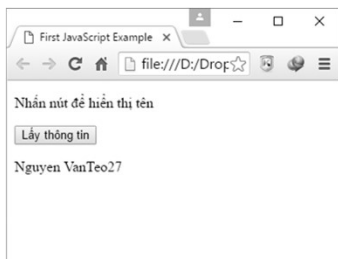
---

---

---

---

## Lệnh for...in



81

---

---

---

---

---

---

---

---

## Câu lệnh with

- Câu lệnh **with** được dùng để thực thi tập hợp các lệnh mà các lệnh này dùng các phương thức của cùng một loại đối tượng.
- Thuộc tính được gán cho đối tượng đã được xác định trong câu lệnh with.
- Cú pháp:

```
with (object) {
 statements;
}
```

82

---

---

---

---

---

---

---

---

## Câu lệnh with

```
<script>
with (document){
 write("This is an example of the things that can be done
");
 write("With the with statment.
");
 write("This can really save some typing");
}
</script>
```

### Kết quả:

This is an example of the things that can be done  
With the with statment  
This can really save some typing

83

---

---

---

---

---

---

---

---

## Câu lệnh with

- with** statement không được khuyến khích sử dụng trong JavaScript hiện đại vì nó có thể gây ra lỗi khó đoán và làm giảm hiệu suất.
- "Nên sử dụng cách truy cập thuộc tính object thông thường để thay thế **with** statement."

84

---

---

---

---

---

---

---

---

## Toán tử new

- Toán tử **new** được dùng để tạo ra một thực thể mới của một loại đối tượng
- Đối tượng có thể có sẵn hoặc do người dùng định nghĩa
 

```
objectName = new objectType(param1 [,param2] ... [,paramN])
```
- Trong đó:
  - **objectName** là tên của thực thể đối tượng mới.
  - **objectType** là một hàm quyết định loại của đối tượng. Ví dụ Array.
  - **Param[1, 2, ...]** là các giá trị thuộc tính của đối tượng.

85

---

---

---

---

---

---

---

---

## Đối tượng string

- Đối tượng **String** được dùng để thao tác và làm việc với chuỗi văn bản.
- Chúng ta có thể tách chuỗi ra thành các chuỗi con và biến đổi chuỗi đó thành các chuỗi hoa hoặc thường trong một chương trình.
- Cú pháp tổng quát:
 

```
stringName.propertyName
```

 hay
 

```
stringName.methodName
```

86

---

---

---

---

---

---

---

---

## Đối tượng string

- Có 3 phương thức khác nhau để tạo ra chuỗi.
  - Dùng lệnh **var** và gán cho nó một giá trị.
 

```
var string = "Test";
```
  - Dùng một toán tử (=) có gán với một tên biến.
 

```
string = "test";
```
  - Dùng hàm khởi tạo **String(string)**

```
string = new String("Test");
```

87

---

---

---

---

---

---

---

---

### Đối tượng Math

- Đối tượng **Math** có các thuộc tính và phương thức biểu thị các phép tính toán học nâng cao.

```
function doCalc(x) {
 var a;
 a = Math.PI * x * x;
 alert ("The area of a circle with a radius of " + x + " is " + a);
}
```

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web

88

88

---

---

---

---

---

---

---

---

### Đối tượng Date

- Mô tả thông tin về: Ngày, Tháng, Năm, giờ, phút, giây của hệ thống.
- Ví dụ: `var now = new Date();`

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web

89

89

---

---

---

---

---

---

---

---

### Các hàm lấy ngày giờ trong đối tượng Date

Tên hàm	Mô tả
getDate()	Ngày: 1..31
getDay()	Ngày trong tuần: 0 (chủ nhật), 1 (thứ 2)
getHours()	Giờ: 0..23
getMinutes()	Phút: 0..59
getMonth()	Tháng: 0 (tháng 1)...11 (tháng 12)
getSeconds()	Giây: 0..59
getTime()	Giờ theo mili giây
getFullYear()	Năm

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web

90

90

---

---

---

---

---

---

---

---

### Ví dụ

```
<script>
 var now= new Date();
 var ngay="";
 ngay="hom nay la ngay"+ now.getDate();
 ngay+=" thang "+now.getMonth();
 ngay+=" nam " + now.getFullYear();
 document.write(ngay) ;
</script>
```



91

---

---

---

---

---

---

---

---

### Bài tập

- Kết quả đoạn code Javascript sau?

```
let x = 5;
let y = "5";
console.log(x == y);
console.log(x === y);
```

**Đáp án:**

true (so sánh bằng, nhưng không xét kiểu dữ liệu)  
false (so sánh cả giá trị và kiểu dữ liệu)



92

---

---

---

---

---

---

---

---

### Bài tập

- Kết quả đoạn code Javascript sau?

```
let sum = 0;
for (let i = 0; i < 5; i++) {
 sum += i;
}
console.log(sum);
```

**Đáp án:**

10 (vì sum = 0 + 1 + 2 + 3 + 4)



93

---

---

---

---

---

---

---

---

## Bài tập

- Kết quả đoạn code Javascript sau?

```
let age = 20;
let result = age >= 18 ? "Lừa hận thù" : "Lừa phần nộ";
console.log(result);
```

**Đáp án:**

"Lừa hận thù" (vì age >= 18 là đúng)

94

---

---

---

---

---

---

---

---

## Bài tập

- Kết quả đoạn code Javascript sau?

```
let numbers = [1, 2, 3, 4, 5];
numbers.splice(2, 1);
console.log(numbers);
```

**Đáp án:**

[1, 2, 4, 5] (vì phần tử tại vị trí 2 bị xóa)

splice(2, 1) sẽ bắt đầu từ vị trí chỉ số 2 của mảng (3 là phần tử ở vị trí chỉ số 2), và sẽ xóa 1 phần tử từ vị trí đó.

95

---

---

---

---

---

---

---

---

## Bài tập

- Kết quả đoạn code Javascript sau?

```
let x = 10;
if (true) {
 let x = 20;
 console.log(x); // ?
}
console.log(x); // ?
```

**Đáp án:**

20 (vì biến x bên trong khối lệnh if là một biến mới).  
10 (vì biến x toàn cục bên ngoài if không bị ảnh hưởng).

96

---

---

---

---

---

---

---

---



## Bài tập

- Kết quả đoạn code Javascript sau?

```
const person = {
 name: "Thanh An",
 getName: function() {
 return this.name;
 }
};

const getPersonName = person.getName;
console.log(getPersonName()); // ?
```

### Đáp án:

**undefined** (vì this trong hàm getPersonName không tham chiếu đến đối tượng person mà tham chiếu đến đối tượng toàn cục window)

97

---

---

---

---

---

---

---

---

## Bài tập

- Viết code Javascript: tính tổng các số chẵn từ 1 đến 100 và in kết quả ra console

```
let sum = 0;
for (let i = 1; i <= 100; i++) {
 if (i % 2 === 0) {
 sum += i;
 }
}
console.log(sum);
```

98

---

---

---

---

---

---

---

---

## Bài tập

- Viết code Javascript: Viết một hàm nhận vào một số nguyên và kiểm tra xem số đó có phải là số nguyên tố hay không. Trả về true nếu là số nguyên tố, ngược lại trả về false.

```
function isPrime(num) {
 if (num < 2) return false;
 for (let i = 2; i <= Math.sqrt(num); i++) {
 if (num % i === 0) return false;
 }
 return true;
}

console.log(isPrime(7)); // true
console.log(isPrime(10)); // false
```

99

---

---

---

---

---

---

---

---

## Bài tập

- Tạo lớp Person và tính tuổi. Yêu cầu: Viết một lớp Person có thuộc tính name, birthYear và phương thức getAge() để tính tuổi dựa trên năm hiện tại.

```
class Person {
 constructor(name, birthYear) {
 this.name = name;
 this.birthYear = birthYear;
 }

 getAge() {
 const currentYear = new Date().getFullYear();
 return currentYear - this.birthYear;
 }
}
```

100

---

---

---

---

---

---

---

---

## Bài tập

- Viết một hàm kiểm tra xem một số có phải là số đối xứng (palindrome) hay không

```
function isPalindrome(num) {
 const str = num.toString();
 return str ===
 str.split('').reverse().join('');
}
```

101

---

---

---

---

---

---

---

---

## Bài tập

- Viết một hàm nhận vào một mảng và trả về một đối tượng đếm số lần xuất hiện của từng phần tử trong mảng.

```
function countOccurrences(arr) {
 return arr.reduce((acc, curr) => {
 acc[curr] = (acc[curr] || 0) + 1;
 return acc;
 }, {});
}
```

102

---

---

---

---

---

---

---

---

JS  
S

## Xử lý sự kiện


Học viện phần mềm và ứng dụng công nghệ thông tin  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 103

103

---

---

---

---

---

---


---

---

JS  
S

## Giới thiệu

- **Event (sự kiện)** là những hành động hoặc thay đổi trạng thái xảy ra trong trang web, ví dụ: người dùng click chuột, di chuyển chuột, submit form, tải trang web, resize cửa sổ,...
- Mục đích: JavaScript cho phép "lắng nghe" (listen) các sự kiện và phản hồi bằng cách thực thi code (event handler).
- Ví dụ:
  - Hiển thị thông báo khi người dùng click vào button.
  - Validate dữ liệu form khi người dùng submit.
  - Tạo hiệu ứng animation khi người dùng di chuyển chuột qua element.


Học viện phần mềm và ứng dụng công nghệ thông tin  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 104

104

---

---

---

---

---

---


---

---

JS  
S

## Các sự kiện thông dụng

- Các sự kiện được hỗ trợ bởi hầu hết các đối tượng
  - onClick
  - onFocus
  - onChange
  - onBlur
  - onMouseOver
  - onMouseOut
  - onMouseDown
  - onMouseUp
  - onLoad
  - onSubmit
  - onResize
  - .....


Học viện phần mềm và ứng dụng công nghệ thông tin  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 105

105

---

---

---

---

---

---

---

---

### Cách thức

- Cách thức

### Gán event handler trực tiếp trong HTML:

<TAG eventHandler="JavaScript Code">

Sử dụng `addEventListener()` method:

```
element.addEventListener(event, handler);
```



Phát triển ứng dụng Web 106

106

## Xử lý sự kiện cho các thẻ HTML

- Cú pháp:

```
<TAG eventHandler = "JavaScript Code">
```

- Ví dụ:

&lt;body&gt;

```
<input type="button" name="Button1" value="OpenSesame!"
```

```
onClick="window.open('mydoc.html');">
```

&lt;/body&gt;

- Lưu ý: Dấu “...” và ‘...’ (nháy kép và nháy đơn)



Phát triển ứng dụng Web 107

107

## Xử lý sự kiện bằng hàm

```
<head>
 <script>
 function GreetingMessage(){
 window.alert("Welcome to my world");
 }
 </script>
</head>
<body onload="GreetingMessage()" >
</body>
```



Phát triển ứng dụng Web 108

108

## Xử lý sự kiện bằng thuộc tính

- Gán tên hàm xử lý cho 1 object event

```
object.eventhandler = function_name;
```

- Ví dụ:

```
<head>
 <script>
 function GreetingMessage(){
 window.alert("Welcome to my world");
 }
 window.onload = GreetingMessage();
 </script>
</head>
```



Phát triển ứng dụng Web 109

109

---

---

---

---

---

---

---

---

## Ví dụ: sự kiện onclick

```
<script>
 function compute(){
 var x = frm.expr.value;
 result.innerHTML = x*x;
 }
</script>
<form name="frm">
 x = <input type="text" name="expr" size=15>

 <input type="button" value="Calculate" onclick="compute()">

 x * x =
</form>
```



Phát triển ứng dụng Web 110

110

---

---

---

---

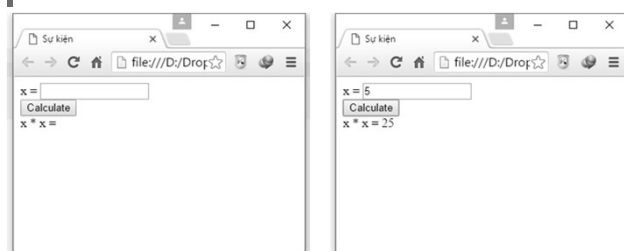
---

---

---

---

## Ví dụ: sự kiện onclick



Phát triển ứng dụng Web 111

111

---

---

---

---

---

---


---

---

### Ví dụ: sự kiện onFocus - onBlur

- Xây ra khi một thành phần HTML nhận focus (onFocus) và mất focus (onBlur)
- Ví dụ:

```
<body bgcolor="lavender">
 <form>
 <input type="text" name="myTextbox"
 onBlur="(document.bgColor='aqua')"
 onFocus="(document.bgColor='dimgray')" />
 </form>
</body>
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UTM-HCM

Phát triển ứng dụng Web 112

112

---

---

---

---

---

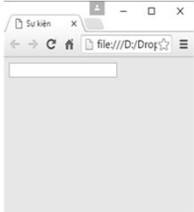
---

---

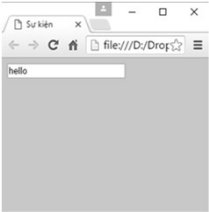
---


### Ví dụ: sự kiện onFocus - onBlur

#### onFocus



#### onBlur





TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UTM-HCM

Phát triển ứng dụng Web 113

113

---

---

---

---

---


---

---

---

### Danh sách sự kiện

Event	Occurs when...
onabort	a user aborts page loading
onblur	a user leaves an object
onchange	a user changes the value of an object
onclick	a user clicks on an object
ondblclick	a user double-clicks on an object
onfocus	a user makes an object active
onkeydown	a keyboard key is on its way down
onkeypress	a keyboard key is pressed
onkeyup	a keyboard key is released



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UTM-HCM

Phát triển ứng dụng Web 114

114

---

---

---

---

---

---

---

---

38

Danh sách sự kiện	
Event	Occurs when...
onload	a page is finished loading.
onmousedown	a user presses a mouse-button
onmousemove	a cursor moves on an object
onmouseover	a cursor moves over an object
onmouseout	a cursor moves off an object
onmouseup	a user releases a mouse-button
onreset	a user resets a form
onselect	a user selects content on a page
onsubmit	a user submits a form
onunload	a user closes a page

115

---

---

---

---

---

---

---

---

---

---

### Quản lý sự kiện

- Sử dụng `addEventListener()` để nâng cao tính linh hoạt và bảo trì mã dễ dàng hơn.
- Ví dụ:  

```
document.getElementById('myButton').addEventListener('click', handleClick);
```

 Thay vì: `<input type="button" value="Click me" onClick="handleClick()">`

116

---

---

---

---

---

---

---

---

---

---

### Quản lý sự kiện

- Event delegation là một kỹ thuật giúp giảm thiểu việc gán sự kiện cho nhiều phần tử con bằng cách gán sự kiện cho phần tử cha và sử dụng cơ chế event bubbling để xử lý sự kiện xảy ra trên các phần tử con. Kỹ thuật này rất quan trọng khi quản lý hiệu suất trên các ứng dụng lớn có nhiều phần tử động.
- Ví dụ:  

```
document.getElementById('parentElement').addEventListener('click', (event) => {
 if (event.target.matches('button')) {
 console.log('Button clicked');
 }
});
```

 Điều này giúp tiết kiệm tài nguyên khi có nhiều phần tử con cần lắng nghe sự kiện.

117

---

---

---

---

---

---

---

---

---

---

JS

S

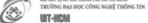
## Quản lý sự kiện

```

<div id="parent">
 <button id="child">Click me</button>
</div>
<script>
document.getElementById("parent").addEventListener('click', () => {
 console.log('Parent clicked');
});
document.getElementById("child").addEventListener('click', () => {
 console.log('Child clicked');
});
</script>

```

Nếu nhấn vào nút (phần tử con), cả hai thông báo "Child clicked" và "Parent clicked" sẽ được in ra do cơ chế event bubbling. Đầu tiên sự kiện sẽ được kích hoạt trên phần tử button và sau đó "bubbling" lên phần tử div cha.



Phát triển ứng dụng Web
118

118

---

---

---

---

---

---

---

---

---

---

JS

S


## Quản lý sự kiện

- Event delegation là một kỹ thuật giúp giảm thiểu việc gán sự kiện cho nhiều phần tử con bằng cách gán sự kiện cho phần tử cha và sử dụng cơ chế event bubbling để xử lý sự kiện xảy ra trên các phần tử con. Kỹ thuật này rất quan trọng khi quản lý hiệu suất trên các ứng dụng lớn có nhiều phần tử động.
- Ví dụ:

```

document.getElementById("parentElement").addEventListener('click', (event) => {
 if (event.target.matches("button")) {
 console.log("Button clicked");
 }
});

```
- Điều này giúp tiết kiệm tài nguyên khi có nhiều phần tử con cần lắng nghe sự kiện.



Phát triển ứng dụng Web
119

119

---

---

---

---

---

---

---

---

---

---

JS

S

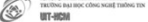
## Ví dụ

- Ví dụ về quản lý form và validate dữ liệu: Kiểm tra dữ liệu nhập vào theo thời gian thực bằng sự kiện input.

```

document.getElementById('email').addEventListener('input', (event) => {
 const emailInput = event.target.value;
 const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
 if (!emailPattern.test(emailInput)) {
 console.log('Invalid email');
 }
});

```



Phát triển ứng dụng Web
120

120

---

---

---

---

---

---

---

---

---

---



### Ví dụ

- Ví dụ về sự kiện bàn phím :

```
document.addEventListener('keydown', (event) => {
 if (event.key === 'ArrowUp') {
 moveElementUp();
 }
});
```

121

---

---

---

---

---

---

---

---

### Ví dụ

- Kết hợp giữa DOM manipulation và event delegation:

```
document.getElementById('myButton').addEventListener('click', function() {
 alert('Button clicked!');
});
```

122

---

---

---

---

---

---

---

---

### Bài tập

- Bài tập 1: Tạo một trang HTML với nút bấm, khi người dùng nhấn vào nút thì hiển thị một thông báo "Nút đã được nhấn!":

```
<button id="myButton">Click me</button>
<script>
 document.getElementById('myButton').addEventListener('click',
function() {
 alert('Nút đã được nhấn!');
});
</script>
```

123

---

---

---

---

---

---

---

---

## Bài tập

- Bài tập 2: Viết một đoạn JavaScript để thay đổi màu nền của một phần tử div khi chuột di chuyển qua phần tử này

```
const div = document.getElementById('myDiv');
div.addEventListener('mouseover', () => {
 div.style.backgroundColor = 'yellow';
});
```

124

---

---

---

---

---

---

---

---

## DOM HTML với Javascript

125

---

---

---

---

---

---

---

---

## Đối tượng HTML DOM

- DOM = **D**ocument **O**bject **M**odel
- Là tập hợp các đối tượng HTML chuẩn được dùng để truy xuất và thay đổi thành phần HTML trong trang web ( thay đổi nội dung tài liệu của trang )
- Một số đối tượng của DOM: window, document, history, link, form, frame, location, event, ...
- Mô hình hóa tài liệu dưới dạng cây, nơi mỗi nút đại diện cho một phần của tài liệu (phần tử, thuộc tính, hoặc văn bản). DOM cho phép lập trình viên truy cập và thay đổi nội dung, cấu trúc và kiểu của tài liệu.

126

---

---

---

---

---

---

---

---


JS

5

## Đối tượng HTML DOM

```
<!DOCTYPE html>
<html>
 <body>
 <h1 id="title">Hello World</h1>
 </body>
</html>
```

DOM sẽ biểu diễn tài liệu này dưới dạng một cấu trúc cây, trong đó html, body, và h1 là các nút (nodes).


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
PTUĐW

Phát triển ứng dụng Web 127

127

---

---

---

---

---

---

---

---


JS

5

## Đối tượng HTML DOM

### Các phương thức thao tác DOM phổ biến

- Truy cập phần tử**
  - document.getElementById(): Trả về phần tử có id cụ thể.
  - document.querySelector(): Trả về phần tử đầu tiên phù hợp với selector CSS.
  - document.querySelectorAll(): Trả về tất cả các phần tử phù hợp với selector CSS.
- Thay đổi nội dung phần tử**
  - innerHTML: Thay đổi nội dung HTML của phần tử.
  - textContent/innerText: Thay đổi nội dung văn bản của phần tử
- Thay đổi thuộc tính của phần tử**
  - setAttribute() và getAttribute(): Thay đổi và lấy giá trị của thuộc tính HTML.


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
PTUĐW

Phát triển ứng dụng Web 128

128

---

---

---

---

---

---

---

---

JS


5

## Đối tượng HTML DOM

**Ví dụ:**  
`const title = document.getElementById('title'); // Truy cập phần tử có id="title"`  
`console.log(title.innerText); // In ra nội dung văn bản bên trong phần tử đó`

**Ví dụ:**  
`document.getElementById('title').innerText = "Welcome to PTUDW Class";`

**Ví dụ:**  
`const img = document.querySelector('img');`  
`img.setAttribute('src', 'image.jpg');`


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
PTUĐW

Phát triển ứng dụng Web 129

129

---

---

---

---

---

---

---

---



JS

5

## Đối tượng Window - DOM

- Là thể hiện của đối tượng cửa sổ trình duyệt
- Tồn tại khi mở 1 tài liệu HTML
- Sử dụng để truy cập thông tin của các đối tượng trên cửa sổ trình duyệt ( tên trình duyệt, phiên bản trình duyệt, thanh tiêu đề, thanh trạng thái ... )

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
133

133

---

---

---

---

---

---

---

---

JS

5

## Đối tượng Window - DOM

<ul style="list-style-type: none"> <li>• Properties <ul style="list-style-type: none"> <li>o document</li> <li>o event</li> <li>o history</li> <li>o location</li> <li>o name</li> <li>o navigator</li> <li>o screen</li> <li>o status</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Methods <ul style="list-style-type: none"> <li>o alert</li> <li>o confirm</li> <li>o prompt</li> <li>o blur</li> <li>o close</li> <li>o focus</li> <li>o open</li> </ul> </li> </ul>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
134

134

---

---

---

---

---

---

---

---

JS

5

## Đối tượng Window - DOM

- Ví dụ:

```
<html>
<body>
<script>
 var curURL = window.location;
 window.alert(curURL);
</script>
</body>
</html>
```

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
135

135

---

---

---

---

---

---

---

---

## Đối tượng Document - DOM



- Biểu diễn cho nội dung trang HTML đang được hiển thị trên trình duyệt
- Dùng để lấy thông tin về tài liệu, các thành phần HTML và xử lý sự kiện

136

---

---

---

---

---

---

---

---

## Đối tượng Document - DOM



- |                                                                                                                                                                                                                                                                                                                                                              |                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• Properties           <ul style="list-style-type: none"> <li>o document</li> <li>o aLinkColor</li> <li>o bgColor</li> <li>o body</li> <li>o fgColor</li> <li>o linkColor</li> <li>o title</li> <li>o URL</li> <li>o vlinkColor</li> <li>o forms[]</li> <li>o images[]</li> <li>o childNodes[]</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>o documentElement</li> <li>o cookie</li> <li>o .....</li> </ul> | <ul style="list-style-type: none"> <li>• Methods           <ul style="list-style-type: none"> <li>o close</li> <li>o open</li> <li>o createTextNode("text")</li> <li>o createElement("HTMLtag")</li> <li>o getElementById("id")</li> <li>o getElementsByName(ten)</li> <li>o getElementsByTagName(Ten_The)</li> <li>o document.tenform.tencontrol.t huoctinh</li> </ul> </li> </ul> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

137

---

---

---

---

---

---

---

---

## Đối tượng Document - DOM



- Truy xuất đến các form:
  - o document.tên\_form
- Truy xuất các đối tượng trong form:
  - o Tên\_form.Tên\_BT
- Thuộc tính đối tượng:
  - o Tên\_thuộc\_tính

138

---

---

---

---

---

---

---

---




JS

5

## Đối tượng Document - DOM

- `appendChild(newNode)`
  - Chèn node mới (**newNode**) vào cuối danh sách các node con của một node
- Ví dụ:

```
<p id="idl" >
 some text
</p>
<script>
 var pNode = document.getElementById("idl");
 var imgNode = document.createElement("img");
 imgNode.src = "images/test.gif";
 pNode.appendChild(imgNode);
</script>
<p id="idl" >
 some text
</p>
```


Học viện quản trị kinh doanh miền Nam  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
142

142

---

---

---

---

---

---

---

---

JS


5

## Đối tượng Document - DOM

- `innerHTML`
  - Chỉ định **nội dung HTML** bên trong một node.
- Ví dụ:

```
<p id="paral" >
 Some text
</p>
<script>
 var theElement = document.getElementById("paral");
 theElement.innerHTML = "Some new text";
</script>
```

Kết quả : Some **new** text


Học viện quản trị kinh doanh miền Nam  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
143

143

---

---

---

---

---

---

---

---

JS


5

## Đối tượng Document - DOM

- `innerText`
  - Tương tự `innerHTML`, tuy nhiên bất kỳ nội dung nào đưa vào cũng được xem như là **text** hơn là các thẻ **HTML**.
- Ví dụ:

```
<script>
 var theElement=document.getElementById("paral");
 theElement.innerText = "Some new text";
</script>
```

// Kết quả hiển thị trên trình duyệt  
"Some <b> new </b> text"


Học viện quản trị kinh doanh miền Nam  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
144

144

---

---

---

---

---

---

---

---



## Đối tượng Document - DOM

- Outer là phần inner và bản thân đối tượng chứa ID.
- Outer gồm có
  - outerHTML lấy nội dung text và tag HTML của cả đối tượng ID
  - outerText : lấy nội dung text
- Inner là nội dung chứa bên trong của đối tượng chứa ID.
- Inner gồm có
  - InnerHTML lấy nội dung text và tag HTML bên trong đối tượng ID
  - innerText: chỉ lấy nội dung text bên trong đối tượng ID

<Div ID=Intro>Monitor<B> SAMSUNG</B></Div>

inner

outer

145

145

## Đối tượng Document - DOM

- s1=Intro.outerText  
 s2=Intro.innerText  
 thì s1 và s2 đều nhận giá trị Monitor SAMSUNG
- s1=Intro.outerHTML  
 s2=Intro.innerHTML  
 Thì s1=<Div Id=Intro>Monitor<B>SAMSUNG</B></Div>  
 Và s2=Monitor<B> SAMSUNG</B>

<Div ID=Intro>Monitor<B> SAMSUNG</B></Div>

inner

outer

146

146

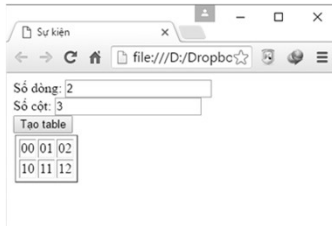
### Một số ví dụ

Phát triển ứng dụng Web 147

147

### Ví dụ: Dynamic table

- Trang web cho phép tạo table có số **dòng**, số **cột** do người dùng nhập.



Phát triển ứng dụng Web 148

148

---

---

---

---

---

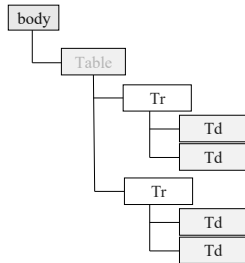
---

---

---

### Ví dụ: Dynamic table

```
<table>
 <tr>
 <td> 12 </td>
 <td> 13 </td>
 </tr>
 <tr>
 <td> 21 </td>
 <td> 22 </td>
 </tr>
</table>
```



149

Phát triển ứng dụng Web

149

---

---

---

---

---

---

---

---

### Ví dụ: Dynamic table

- `document.createElement (...)` :Tạo một đối tượng thẻ DOM HTML
- `object.appendChild (...)` : Thêm một đối tượng thẻ DOM HTML như là nút con.



Phát triển ứng dụng Web 150

150

---

---

---

---

---

---

---

---

### Ví dụ: Dynamic table (1)



```
<body>
<form name="frm">
Số dòng: <input type="text" name="txtdong">

Số cột: <input type="text" name="txtcot">

<input type="button" name="taoTable" value="Tao table"
onclick="createTable(document.getElementById('divTable'))">
</form>
<div id="divTable">
</div>
</body>
```

151

---

---

---

---

---

---

---

---

---

---

### Ví dụ: Dynamic table (1)



```
function createTable(divTable){
 var tagTable = document.createElement("table");
 tagTable.border=1;
 var nDong = frm.txtdong.value;
 var nCot = frm.txtcot.value;
 for (i=0;i<nDong;i++){
 var tr = document.createElement("tr");
 for (j=0;j<nCot;j++){
 var td = document.createElement("td");
 var textTd = document.createTextNode(i+" "+j);
 td.appendChild(textTd);
 tr.appendChild(td);
 }
 tagTable.appendChild(tr);
 }
 divTable.appendChild(tagTable);
}
```

152

---

---

---

---

---

---

---

---

---

---

### Ví dụ: Dynamic table (2)



```
<body>
<form name="frm">
Số dòng: <input type="text" name="txtdong">

Số cột: <input type="text" name="txtcot">

<input type="button" name="taoTable" value="Tao table" onclick="createTable()">
</form>
<table id="divTable">
</table>
</body>
```

153

---

---

---

---

---

---

---

---

---

---

### Ví dụ: Dynamic table (2)

```
function createTable(){
 var tagTable = document.getElementById("divTable");
 tagTable.border = 3;
 var nDong = frm.txtdong.value;
 var nCot = frm.txtcot.value;
 for (i=0;i<nDong;i++){
 var tr = document.createElement("tr");
 for (j=0;j<nCot;j++){
 var td = document.createElement("td");
 var textTd = document.createTextNode(i+" "+j);
 td.appendChild(textTd);
 tr.appendChild(td);
 }
 tagTable.appendChild(tr);
 }
}
```



Phát triển ứng dụng Web 154

154

---

---

---

---

---

---

---

---

### Ví dụ

- Hiệu ứng chữ chạy trong trang web



Phát triển ứng dụng Web 155

155

---

---

---

---

---

---

---

---

### Lý thuyết

- Lệnh **setTimeout(f, n)** quy định sau khoảng thời gian n mili giây hàm f sẽ được gọi. (f là chuỗi lưu lệnh cần thực hiện)
- Vài lệnh khác cùng nhóm setTimeout
  - **timeID = setTimeout(f, n)**
  - **clearTimeout(timeID)**: Hủy setTimeout
  - **intervalID = setInterval(f, n)**: Sau mỗi khoảng thời gian n ms lệnh f được gọi và tiếp tục cho đến khi thoát chương trình
  - **clearInterval(intervalID)**: Hủy interval.
- Giả sử str là một chuỗi ta có
  - **str.length**: Thuộc tính cho biết độ dài chuỗi
  - **str.substr(i, n)**: lấy ra n ký tự kể từ vị trí thứ i (Ký tự đầu tiên được đánh số là 0)



Phát triển ứng dụng Web 156

156

---

---

---

---

---

---

---

---

## Giải thuật

- Ý tưởng giải thuật
  - Để có được cảm giác chữ chạy trong một thẻ <div> hoặc <p> ta cần copy ký tự đầu của dòng chữ hiện tại đưa xuống cuối cùng và lặp lại như vậy sau mỗi khoảng thời gian.
- Giải thuật: Giả sử ta có biến str đang lưu chuỗi cần chạy. Công việc thực hiện như sau:
  - **B1:** Thể hiện chuỗi str trong thẻ <div>. Chuyển sang bước 2
  - **B2:** Chuyển ký tự đầu của **str** về cuối (bằng cách gán str = xâu con kể từ vị trí thứ 2 của str đến cuối + ký tự đầu tiên của str). Chuyển sang bước 3
  - **B3:** Trễ một khoảng thời gian rồi quay lại bước 1



Phát triển ứng dụng Web 157

157

---

---

---

---

---

---

---

---

## Mã lệnh

```
<script>
 var str= 'Đại Học Công Nghệ Thông Tin';
 for (i=str.length; i<100; i++){
 str = str + ' ';
 }

 function ChuChay(){
 var tagDiv = document.getElementById("chaychu");
 tagDiv.innerText = str;
 str = str.substr(1,str.length-1) + str.charAt(0);
 setTimeout(ChuChay,100);
 }
</script>
<body onload="ChuChay()">
<div id="chaychu"> </div>
</body>
```



Phát triển ứng dụng Web 158

158

---

---

---

---

---

---

---

---

## Phát triển

- Thay bằng nhiều dòng chữ chạy khác nhau (sử dụng mảng để lưu trữ)
- Chữ chạy theo nhiều cách khác nhau
- Cho chữ chạy trên thanh tiêu đề (dùng **document.title**)
- Cho chữ chạy trên một đối tượng khác



Phát triển ứng dụng Web 159

159

---

---

---

---

---

---

---

---

## Bài tập

- **Bài tập 2:** Viết JavaScript để thay đổi nội dung của một phần tử h1 có id="header" thành "Chào mừng bạn đến với lớp học!".

```
document.getElementById('header').innerText = 'Chào mừng bạn đến với lớp học!';
```

160

---

---

---

---

---

---

---

---

## Bài tập

- **Bài tập 1:** Viết JavaScript để thêm một phần tử p mới với nội dung "Hello World" vào cuối body.

```
const newParagraph = document.createElement('p');
newParagraph.textContent = 'Hello World';
document.body.appendChild(newParagraph);
```

161

---

---

---

---

---

---

---

---

## Cơ chế Xử lý Bất đồng bộ trong JS

162

---

---

---

---

---


---


---

---

## Xử lý Đồng bộ và Bất đồng bộ

- Xử lý đồng bộ: Synchronous
- Xử lý bất đồng bộ: Asynchronous








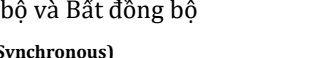
163

[illegible]

## Xử lý Đồng bộ và Bất đồng bộ

- **Xử lý đồng bộ (Synchronous)**
- Thực hiện các công việc một cách tuần tự, công việc này xong thì mới được thực hiện các công việc khác.
- Ví dụ có 2 công việc A và B thì khi có nghĩa là A thực hiện xong trước rồi mới tới lượt B.





Logo thương hiệu của trường và các đơn vị  
THƯỜNG BAN HỌ TÊN CÔNG NGHỆ THÔNG TIN


UUT-HOCH

Phát triển ứng dụng Web

164

164

[illegible]



## Xử lý Đồng bộ và Bất đồng bộ

- Điều này sẽ ảnh hưởng đến hiệu suất của người dùng.
- Giả sử một request gửi lên server yêu cầu server thực hiện chức năng như import file hoặc đọc ghi file thì lúc này server sẽ mất nhiều thời gian để xử lý những việc này. Đồng nghĩa với việc trong lúc server thực hiện chức năng đó thì sẽ không thể thực hiện thêm một hành động nào khác.

165

---

---

---

---

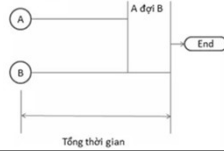
---

---

## Xử lý Đồng bộ và Bất đồng bộ

### • Multi-thread

- Để khắc phục tình trạng nói trên, các ngôn ngữ lập trình như C/C++, Java,... sẽ sử dụng cơ chế đa luồng (multi-thread). Nghĩa là mỗi công việc tốn thời gian sẽ được thực hiện trên một thread riêng biệt mà không can thiệp vào thread chính. Vẫn có thể thực hiện các công việc tốn thời gian mà vẫn có thể bắt các sự kiện ở thread chính.



Phát triển ứng dụng Web 166

166

---

---

---

---

---

---

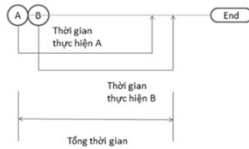
---

---

## Xử lý Đồng bộ và Bất đồng bộ

### • Xử lý bất đồng bộ: Asynchronous

- Javascript là ngôn ngữ Single thread (đơn luồng)
- Với cách xử lý bất đồng bộ, khi A bắt đầu thực hiện, chương trình tiếp tục thực hiện B mà không đợi A kết thúc. Chúng ta sẽ cung cấp một phương thức để chương trình thực hiện khi A hoặc B kết thúc.



Phát triển ứng dụng Web 167

167

---

---

---

---

---

---

---

---

## Xử lý Đồng bộ và Bất đồng bộ

- Cơ chế thực hiện việc này trong JavaScript có thể là sử dụng
  - Callback
  - Promise (ES6)
  - Async/await (ES6)

168

---

---

---

---

---

---

---

---



## Xử lý Đồng bộ và Bất đồng bộ

- Callback:** là một hàm được truyền vào một hàm khác như một tham số, và sẽ được thực thi sau khi hàm đó hoàn thành. Callback thường được sử dụng để xử lý các tác vụ bất đồng bộ (asynchronous), ví dụ như đọc file, gửi request HTTP, setTimeout,...
- Ví dụ:

```
function readFile(filename, callback) {
 // Đọc file bất đồng bộ
 // ...
 // Khi đọc file xong, gọi hàm callback với kết quả
 callback(data);
}
readFile('myfile.txt', function(data) {
 console.log(data);
});
```

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
169

169

---

---

---

---

---

---

---

---

## Xử lý Đồng bộ và Bất đồng bộ

- Callback:**
- Ưu điểm: Đơn giản, dễ hiểu, được hỗ trợ từ lâu trong Javascript.
- Nhược điểm:
  - Callback Hell: Khi có nhiều tác vụ bất đồng bộ lồng nhau, code sẽ trở nên khó đọc và khó bảo trì (nhiều callback lồng nhau).
  - Khó quản lý lỗi: Xử lý lỗi trong callback phức tạp hơn.

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
170

170

---

---

---

---

---

---

---

---

## Xử lý Đồng bộ và Bất đồng bộ

- Promise:** là một đối tượng đại diện cho kết quả của một tác vụ bất đồng bộ, có thể thành công (resolve) hoặc thất bại (reject). Promise giúp viết code asynchronous dễ đọc và dễ quản lý hơn callback
- Trạng thái của Promise:
  - Pending: Tác vụ bất đồng bộ đang thực thi. Promise đang chờ hoàn thành.
  - Fulfilled (Resolved): Tác vụ (Promise) hoàn thành thành công.
  - Rejected: Tác vụ thất bại.
- Promise chain là một chuỗi các promise được liên kết với nhau bằng phương thức .then(). Khi một promise được resolve, nó sẽ gọi hàm .then() tiếp theo trong chuỗi, và truyền kết quả của promise đó vào hàm .then().

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web
171

171

---

---

---

---

---

---

---



---


## Xử lý Đồng bộ và Bất đồng bộ

- **Promise:**
- Ví dụ:

```
function readFile(filename) {
 return new Promise((resolve, reject) => {
 // Đọc file bất đồng bộ
 // ...
 // Nếu đọc file thành công
 resolve(data);
 // Nếu đọc file thất bại
 reject(error);
 });
}

readFile('myfile.txt')
 .then(data => console.log(data))
 .catch(error => console.error(error));
```



học tập - ứng dụng của React Native và các công nghệ mới nhất

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



UIT-HCM

**Phát triển ứng dụng Web**

172


172

[illegible]

## Xử lý Đồng bộ và Bất đồng bộ

- Promise:**
- Ưu điểm:**
  - Dễ đọc, dễ quản lý: Sử dụng then và catch để xử lý kết quả và lỗi, tránh callback hell.
  - Quản lý lỗi tốt hơn: Catch block xử lý tất cả lỗi trong promise chain.
- Nhược điểm:** Cú pháp phức tạp hơn callback.



Đơn vị phát triển của hệ thống quản lý tài sản  
 VÀ CÁC DỊCH VỤ LIÊN QUAN  
 VIỆT - COM

Phát triển ứng dụng Web



173


173

[illegible]

## Xử lý Đồng bộ và Bất đồng bộ

- **Async/Await**: là cú pháp "đường" (syntactic sugar) được xây dựng dựa trên Promise, giúp viết code asynchronous trông giống code synchronous, dễ đọc và dễ hiểu hơn.
- Cách sử dụng:
  - async: Khai báo hàm asynchronous, cho phép sử dụng await bên trong hàm.
  - await: Dừng thực thi code cho đến khi Promise được resolve, sau đó trả về kết quả.
- Syntactic sugar là cú pháp được thêm vào ngôn ngữ lập trình để làm cho code dễ đọc và dễ viết hơn, mà không thay đổi khả năng cơ bản của ngôn ngữ. Ví dụ: Arrow functions.



Khoa Học Công Nghệ và Truyền Thông Số

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

**Phát triển ứng dụng Web**

174

174

---


---

---

---

---

---




## Xử lý Đồng bộ và Bất đồng bộ

- **Async/Await: (từ ES8)**
- Ví dụ:
 

```
async function readAndLog() {
 try {
 const data = await readFile('myfile.txt');
 console.log(data);
 } catch (error) {
 console.error(error);
 }
}

readAndLog();
```


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 175

175

---

---

---


---

---

---


---

---



## Xử lý Đồng bộ và Bất đồng bộ

- **Async/Await:**
- Ưu điểm:
  - Cú pháp đơn giản, dễ đọc: Code asynchronous trông giống code synchronous.
  - Dễ dàng quản lý lỗi: Sử dụng try/catch để xử lý lỗi.
- Nhược điểm: Yêu cầu sử dụng Promise, chỉ hoạt động trong hàm async.


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 176

176

---

---

---


---

---

---

---


---



## Xử lý Đồng bộ và Bất đồng bộ

- **setTimeout() và setInterval():**
- setTimeout() và setInterval() đều là các hàm xử lý thời gian trong JavaScript.
- setTimeout(): Thực hiện một hành động sau một khoảng thời gian nhất định (chỉ thực hiện một lần sau khi thời gian trôi qua).
- Ví dụ: setTimeout(function, milliseconds);
- setTimeout(() => {
 

```
 console.log("Hello after 2 seconds");
 }, 2000); // Thực hiện sau 2 giây
```
- clearTimeout(): Dùng để hủy bỏ hành động đã được thiết lập bằng setTimeout()


TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 177

177

---

---

---

---

---

---

---

---

## Xử lý Đồng bộ và Bất đồng bộ

- **setTimeout()** và **setInterval()**:
- setTimeout() và setInterval() đều là các hàm xử lý thời gian trong JavaScript.
- setInterval(): Thực hiện một hành động **lặp đi lặp lại** sau mỗi khoảng thời gian nhất định
- Ví dụ: `setInterval(function, milliseconds);`  
`setInterval(() => {`  
`console.log("Hello every 2 seconds");`  
`}, 2000);` // Thực hiện mỗi 2 giây
- **clearInterval()**: Dùng để dừng việc lặp lại hành động đã được thiết lập bằng setInterval(). Điều này ngăn không cho hàm tiếp tục được thực hiện theo chu kỳ lặp lại.



Phát triển ứng dụng Web 178

178

---

---

---

---

---

---

---

---

## ES6



Phát triển ứng dụng Web 179

179

---

---

---

---

---

---

---

---

## Vanilla JavaScript

- **Vanilla JavaScript** là thuật ngữ dùng để chỉ JavaScript thuần, không phụ thuộc vào bất kỳ thư viện hoặc framework nào như jQuery, React, Vue.js,...
- Đây là cách lập trình JavaScript trực tiếp bằng chính ngôn ngữ JavaScript mà không cần thêm bất kỳ công cụ nào bên ngoài.
- Ưu điểm: Hiệu năng. Tính tương thích cao. Sử dụng trực tiếp.
- JavaScript đã trải qua nhiều phiên bản phát triển dựa trên đặc tả ECMAScript (ES)



Phát triển ứng dụng Web 180

180

---

---

---

---


---

---

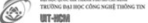
---

---

ES6



- JavaScript ES6 (EcmaScript 6) là một phiên bản của ngôn ngữ lập trình Javascript. ES6 được đưa ra vào năm 2015 nhằm cải thiện và bổ sung thêm tính năng cho Javascript.
- ES6 bao gồm nhiều tính năng mới và cải tiến, từ cú pháp đến cách làm việc với các đối tượng trong Javascript. Những tính năng này giúp cho việc lập trình trở nên dễ dàng hơn và tăng tính tương thích cho mã nguồn khi sử dụng trên các trình duyệt hiện đại. Hiện nay, Javascript ES6 đã được hỗ trợ rộng rãi trên các trình duyệt và đang dần trở thành chuẩn mới cho ngôn ngữ lập trình Javascript.



Phát triển ứng dụng Web
181

181

---

---

---

---


---

---

---


---

ES6



- Đã có trên 14 phiên bản ECMAScript (ES) được phát hành (bản ES4.0 bị hủy bỏ)
- Ví dụ:
 

ES6	2015	ES2015 ...
ES9	2018	ES2018 ...
ES15	2024	ES2024
- Mỗi phiên bản ES mới đều bổ sung thêm các tính năng và cải tiến cho JavaScript.
- Các trình duyệt hiện đại hỗ trợ hầu hết các tính năng của ES6+, nhưng có thể cần sử dụng transpiler (ví dụ: Babel) để chuyển đổi code ES6+ sang code ES5, tương thích với các trình duyệt cũ hơn.



Phát triển ứng dụng Web
182

182

---

---

---

---


---

---

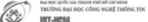
---

---

ES6



- Những tính năng mới và cải tiến trong ES6 bao gồm:
  - Khai báo biến với let và const
  - Arrow functions
  - Template literals
  - Destructuring
  - Rest parameters
  - Spread syntax
  - Classes



Phát triển ứng dụng Web
183

183

---

---

---

---

---

---

---

---

## ES6

- **Arrow Functions (Hàm mũi tên):**

- Giúp khai báo hàm ngắn gọn và có cú pháp rõ ràng. Không có phạm vi this riêng, phù hợp với xử lý bất đồng bộ và callback.

```
// Cú pháp thông thường
function sum(a, b) {
 return a + b;
}
// Arrow function
const sum = (a, b) => a + b;
```



Phát triển ứng dụng Web 184

184

## ES6

- **Arrow Functions (Hàm mũi tên):**

```
// Arrow function với một tham số
const square = (x) => x * x;
console.log(square(4)); // 16
```

Hoặc:

```
const greet = (name) => `Hello, ${name}!`;
console.log(greet("Alice")); // "Hello, Alice!"
```



Phát triển ứng dụng Web 185

185

## ES6

- **Template Literals (Chuỗi mẫu):**

- Sử dụng dấu backtick ` , cho phép chèn biến vào chuỗi một cách dễ dàng và hỗ trợ xuống dòng.


```
const name = 'Mỹ Lan';
const age = 60;
const message = `Chào, ${name}! Welcome to ES6.`;
console.log(message);
const message = `Tên tôi là ${name}. Tôi ${age} tuổi.`;
console.log(message); // Tên tôi là Mỹ Lan. Tôi 60 tuổi.
```



Phát triển ứng dụng Web 186

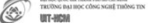
186

ES6



- Destructuring (Phân rã):**
  - Trích xuất các giá trị từ mảng hoặc đối tượng và gán chúng vào các biến riêng biệt, giúp mã nguồn ngắn gọn và dễ hiểu hơn..

```
// Phân rã mảng
const [first, second] = [1, 2];
console.log(first); // 1
Hoặc:
const array = [1, 2, 3];
const [first, second, third] = array;
console.log(first, second, third); // 1 2 3
```



Phát triển ứng dụng Web
187

187

---

---

---

---


---

---

---

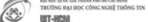
---

ES6



- Destructuring (Phân rã):**
  - Phân rã đối tượng

```
// Phân rã đối tượng
const { name, age } = { name: 'Alice', age: 25 };
console.log(name); // Alice
Hoặc:
const person = { name: "Alice", age: 25 };
const { name, age } = person;
console.log(name); // Alice
console.log(age); // 25
```



Phát triển ứng dụng Web
188

188

---

---

---

---


---

---

---

---

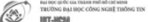
ES6



- Default Parameters (Tham số mặc định):**
  - Cho phép đặt giá trị mặc định cho các tham số khi không được truyền vào hàm.

```
function greet(name = "Guest") {
 return `Hello, ${name}!`;
}

console.log(greet()); // "Hello, Guest!"
console.log(greet("Mỹ Lan")); // "Hello, Mỹ Lan!"
```



Phát triển ứng dụng Web
189

189

---

---

---

---

---

---

---

---





### Một số cải tiến của các phiên bản sau ES6

- **ES7: Exponentiation Operator:** Toán tử lũy thừa (\*\*) để thay thế cho Math.pow()

```
console.log(2 ** 3); // 8
```

- **Array.includes():** Kiểm tra xem một mảng có chứa phần tử nào đó hay không.

```
let arr = [1, 2, 3];
console.log(arr.includes(2)); // true
```

193

---

---

---

---

---

---

---

---

### Một số cải tiến của các phiên bản sau ES6

- **ES11: Optional Chaining (?):** Giúp truy cập các thuộc tính sâu mà không lo gặp lỗi undefined.

```
let user = { address: { city: "Hanoi" } };
console.log(user?.address?.city); // "Hanoi"
console.log(user?.contact?.phone); // undefined
```

- **Nullish Coalescing Operator (??):** Toán tử trả về giá trị phía bên phải nếu giá trị phía bên trái là null hoặc undefined.

```
let x = null;
let result = x ?? "default";
console.log(result); // "default"
```

194

---

---

---

---

---

---

---

---

### Sử dụng Vanilla JavaScript thay vì thư viện hoặc framework

- **Dự án nhỏ và vừa:** Với các dự án web có yêu cầu đơn giản hoặc vừa phải, Vanilla JS là lựa chọn lý tưởng vì giúp giảm bớt tải trọng trang web và đơn giản hóa mã nguồn.
- **Hiệu năng:** Khi cần tối ưu hóa hiệu suất và không cần tải thêm thư viện như jQuery hoặc React.
- **Tương thích trình duyệt:** JavaScript hiện đại đã hỗ trợ rất tốt trên hầu hết các trình duyệt, không cần phải lo lắng về sự tương thích như trước.

195

---

---

---

---

---

---

---

---

JS  
5

## Biểu thức chính quy Regular Expression (Regex)

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 196

196

---

---

---

---

---

---

---

---

JS  
5

## Biểu thức chính quy

- **Regular Expressions (RegEx, regexp hay regxp)** là một công cụ mạnh mẽ để xử lý và kiểm tra chuỗi ký tự, giúp tìm kiếm, thay thế hoặc kiểm tra mẫu chuỗi một cách dễ dàng và nhanh chóng, chính xác.
- Là một chuỗi miêu tả một bộ các chuỗi khác, theo những quy tắc cú pháp nhất định
- Thường được dùng trong các trình biên tập văn bản và các tiện ích tìm kiếm và xử lý văn bản dựa trên các mẫu được quy định
- Các tính năng này rất hữu dụng trong việc kiểm tra dữ liệu người dùng nhập vào các ô nhập liệu
- **Lưu ý:** việc kiểm tra dữ liệu tại client chỉ nhằm tăng tính thuận tiện cho người dùng và giảm lưu lượng xử lý trên server.

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 197

197

---

---

---

---

---

---

---

---

JS  
5

## Biểu thức chính quy

- **Cú pháp cơ bản của Regular Expressions:**
- Khai báo:
  1. var bienRegex = /pattern/attributes;
  2. var reExample = new RegExp("pattern",attributes);

Trong đó:

- pattern là mẫu so khớp trong biểu thức chính quy.
- attributes xác định bằng các giá trị:

"g" (áp dụng toàn diện, tất là tìm tất cả các chuỗi con so khớp với mẫu;  
 "i" (case-insensitive: phân biệt hoa-thường);  
 "m" (áp dụng với đoạn văn bản nhiều dòng).

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 198

198

---

---

---

---

---

---

---

---



## Biểu thức chính quy

- Ký tự đặc biệt và phạm vi:
  - `^`: Bắt đầu một chuỗi. Ví dụ: `/^php(.*?)` : những chuỗi bắt đầu bằng chữ php
  - `$`: Kết thúc một chuỗi. Ví dụ: `/(.*)basic$/` : những chuỗi kết thúc bằng chữ basic
  - `[abc]`: Khớp với bất kỳ ký tự nào trong abc.
  - `[^abc]`: Khớp với bất kỳ ký tự nào không phải abc.
  - `[a-z]`: Khớp với bất kỳ ký tự nào từ a đến z.
  - `/a|b/` : ký tự a hoặc b
  - `/ab?c/` : chuỗi có dạng abc hoặc ac. Còn `()` như `/(ab)+/` (khớp "ab", "abab",...)



Phát triển ứng dụng Web 202

202

---

---

---

---

---

---

---

---

## Biểu thức chính quy

- Lặp lại và nhóm - Quantifiers (Bộ định lượng):
  - `*`: Lặp lại từ 0 lần trở lên. Ví dụ: `/a*/` : `<null>`, a, aa, aaa,.....
  - `+`: Lặp lại từ 1 lần trở lên. Ví dụ: `/a+/` : a,aa,aaa,.....
  - `?`: Tùy chọn, xuất hiện hoặc không xuất hiện. `/colou?r/` (khớp "color" hoặc "colour")
  - `{n}`: Xuất hiện đúng n lần. Ví dụ: `/\d{3}/` (khớp "123", "456",...)
  - `{n,}`: Xuất hiện ít nhất n lần. `/\d{2,}/` (khớp "12", "123",...)
  - `{n,m}`: Xuất hiện từ n đến m lần. Ví dụ: `/a{1,3}/` : a,aa,aaa



Phát triển ứng dụng Web 203

203

---

---

---

---

---

---

---

---

## Biểu thức chính quy

- Ví dụ: Kiểm tra định dạng email:
 

```
const emailPattern = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
const email = "example@example.com";
console.log(emailPattern.test(email)); // true
```



Phát triển ứng dụng Web 204

204

---

---

---

---

---

---

---

---

### Biểu thức chính quy

- Ví dụ: Tìm kiếm số điện thoại:

```
const phonePattern = /\d{3}-\d{3}-\d{4}/;
const text = "Liên hệ: 123-456-7890";
console.log(text.match(phonePattern)); // ["123-456-7890"]
```



Phát triển ứng dụng Web 205

205

---

---

---

---

---

---

---

---

### Biểu thức chính quy

- Ví dụ: Kiểm tra số điện thoại (Việt Nam):

```
const phoneRegex = /^(0|84)(\s|.)?([32-9])(5[689])|(7[06-9])(8[1-689])|(9[0-46-9]))(\d)(\s|.)?(\d{3})(\s|.)?(\d{3})/;
console.log(phoneRegex.test("0987654321")); // true
console.log(phoneRegex.test("123-456-789")); // false
```



Phát triển ứng dụng Web 206

206

---

---

---

---

---

---

---

---

### Biểu thức chính quy

- Ví dụ: Tìm kiếm và thay thế chuỗi trong văn bản:

```
const text = "Tôi có 1 quả táo và 1 cây bút.";
const newText = text.replace(/\d+/g, "một số");
console.log(newText); // "Tôi có một số quả táo và một số cây bút."
```



Phát triển ứng dụng Web 207

207

---

---

---

---

---

---

---

---

## Biểu thức chính quy



- Tính năng đặc biệt của Regular Expressions trong Javascript:
  - Phương thức `test()`: Kiểm tra xem chuỗi có khớp với mẫu hay không. Nếu có hàm trả về giá trị `true`, ngược lại trả về `false`.
  - Phương thức `match()`: Tìm và trả về các phần tử khớp với mẫu.
  - Phương thức `replace()`: Thay thế các phần tử khớp với mẫu bằng chuỗi mới.
  - Phương thức `exec()`: Kiểm tra xem chuỗi có khớp với mẫu hay không. Nếu có hàm trả về mảng các chuỗi so khớp, nếu không trả về `null`.
  - Phương thức khác: `search()`, `split()`...

208

---

---

---

---

---

---

---

---

## Biểu thức chính quy



- Ví dụ: kiểm tra biến `name` có chứa 1 chữ `b` hoặc 1 chữ `t`

```
var name = "Bob";
re = /[bt]/;

if(re.test(name)){
 alert ('FOUND A b OR t IN NAME');
}else{
 alert ('DID NOT FIND A b OR t IN NAME');
}
```

209

---

---

---

---

---

---

---

---

## Biểu thức chính quy



- Ví dụ: Kiểm tra xem chuỗi `input` có phải là mã số sinh viên hợp lệ (bắt đầu bằng 'SV' và theo sau là 7 chữ số) hay không:

```
const studentIdRegex = /^SV\d{7}$/;
console.log(studentIdRegex.test("SV1234567")); // true
console.log(studentIdRegex.test("SV123456")); // false
```

210

---

---

---

---

---

---

---

---

## Biểu thức chính quy



- Ví dụ: Xác thực mật khẩu mạnh. Biểu thức chính quy (ít nhất 8 ký tự, gồm chữ cái thường, chữ hoa, số và ký tự đặc biệt):

```
const passwordPattern = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*~`~\W_]).{8,}$/;
```

```
const password = "P@ssw0rd!";
```

```
console.log(passwordPattern.test(password)); // true
```



Phát triển ứng dụng Web 211

211

---

---

---

---

---

---

---

---

## Q & A



**Cảm ơn đã theo dõi**

Hãy vọng cùng nhau đi đến thành công.



Phát triển ứng dụng Web 212

212

---

---

---

---

---

---

---

---