Edison Technology
Academy
Learning by Doing

JavaScript cơ bản

Khóa học ReatJs

Mục tiêu bài học

Edison Technology Academy
Learning by Doing

- Nắm được tổng quan về JavaScript
- Tìm hiểu các kiểu dữ liệu trong JavaScript
- Toán tử trong Javascript
- Cấu trúc điều khiển trong JavaScript
- Vòng lặp trong JavaScript

Edison Technology
Academy
Learning by Doing

Tổng quan về JavaScript

Giới thiệu tổng quan

Edison Technology Academy
Learning by Doing

- JavaScript là một ngôn ngữ lập trình nhẹ, đa nền tảng.
- Vừa là kiểu ngôn ngữ mệnh lệnh vừa là kiểu ngôn ngữ khai báo.
- Có một thư viện tiêu chuẩn của các đối tượng, như Mảng, Ngày tháng và Toán học, và một tập hợp cốt lõi của các phần tử ngôn ngữ như toán tử, cấu trúc điều khiển và câu lệnh.
- Sử dụng cho các ứng dụng phát triển theo hướng Máy khách và Máy chủ.

Cú pháp

Edison Technology Academy
Learning by Doing

Cú pháp:

```
<script>

// JavaScript Code

</script>
```

Ví dụ:

```
<body>

    <script>

        console.log("Welcome to JavaScript!");

    </script>

</body>

</html>
```

Kết quả: Hiển thị ở màn hình console
Welcome to JavaScript!

Tính năng của JavaScript

Edison Technology Academy
Learning by Doing

- JavaScript đã được tạo ra để thao tác với DOM, giúp biến các trang web tĩnh thành web động.
- Các hàm trong JavaScript là các đối tượng với các thuộc tính và phương thức. Chúng có thể được truyền dưới dạng đối số trong các hàm khác.
- Có thể xử lý ngày và giờ.
- Thực hiện xác thực biểu mẫu.
- Không cần trình biên dịch.

Ứng dụng của JavaScript

Edison Technology Academy
Learning by Doing

- Phát triển website: Thêm tính tương tác và hành vi cho các trang web tĩnh.
- Ứng dụng web: Hỗ trợ phát triển ứng dụng trên nền tảng web.
- Ứng dụng máy chủ: Giúp phát triển các tác vụ phía máy chủ.
- Trò chơi: Sự kết hợp của JavaScript và HTML5 làm cho JavaScript cũng trở nên phổ biến trong phát triển trò chơi.
- Ứng dụng di động: JavaScript cũng có thể được sử dụng để tạo ứng dụng di động.

Hạn chế của JavaScript

Edison Technology Academy
Learning by Doing

- Rủi ro bảo mật: JavaScript có thể được sử dụng để lấy dữ liệu bằng AJAX hoặc bằng cách thao tác với các thẻ tải dữ liệu như <img>, <object>, <script>. Kiểu tấn công này được gọi là Cross Site Script.
- Hiệu suất: JavaScript không cung cấp cùng mức hiệu suất như nhiều ngôn ngữ truyền thống cung cấp vì một chương trình phức tạp được viết bằng JavaScript sẽ tương đối chậm.
- Độ phức tạp: Để thành thạo một ngôn ngữ script, phải có kiến thức toàn diện về tất cả các khái niệm lập trình, các đối tượng ngôn ngữ cốt lõi, các đối tượng phía máy khách và phía máy chủ.
- Khả năng kiểm tra kiểu dữ liệu của biến và xử lý lỗi yếu.

Edison Technology
Academy
Learning by Doing

Cách khai báo biến trong JavaScript

Global variable

Edison Technology Academy
Learning by Doing

- là các biến được khai báo và sử dụng trên toàn bộ phạm vi của chương trình JavaScript

```
var helloWorld1 = "Hello world 1";
let helloWorld2 = "Hello world 2";
const helloWorld3 = "Hello world 3";
```

Scope variable

Edison Technology Academy
Learning by Doing

- Tương tự như global variable thì nó được sử dụng để khai báo biến, tuy nhiên nó được khai báo trong 1 phạm vi cụ thể như trong 1 function, object ...
- Các biến này chỉ được sử dụng trong phạm vi của function hay object đó.
- Chỉ có thể sử dụng let hoặc const để khai báo biến.

```
function getName() {
  let name = "My name is ABC"

  return name
}
```

Edison Technology
Academy
Learning by Doing

Các kiểu dữ liệu trong JavaScript

JavaScript là Dynamic type

Edison Technology Academy
Learning by Doing

Dynamic type: các biến không được liên kết trực tiếp với bất kỳ loại giá trị cụ thể nào và bất kỳ biến nào cũng có thể được gán (và gán lại) giá trị của tất cả các loại:

```
let foo = 10; // foo là một số

foo = "apple"; // foo là một chuỗi

foo = true; // foo là một giá trị boolean
```

JavaScript là Weak type

Edison Technology Academy
Learning by Doing

Weak type: cho phép chuyển đổi kiểu ngầm khi một thao tác liên quan đến các kiểu không khớp, thay vì đưa ra lỗi:

```
const foo = 123; // foo là một số

const result = foo + "4"; // JavaScript ép kiểu biến foo thành một chuỗi

console.log(result); // 1234
```

Kiểu dữ liệu nguyên thủy

Edison Technology Academy
Learning by Doing

Giá trị nguyên thủy (Primitive values)

- Kiểu Boolean: là kiểu dữ liệu logic chỉ có thể có các giá trị đúng hoặc sai.
- Kiểu Null: thể hiện giá trị rỗng hoặc không tồn tại.
- Kiểu Undefined: biến chưa được gán giá trị có giá trị là undefined.
- Kiểu Number: lưu trữ số

  NaN ("Not a Number") thường gặp khi kết quả của một phép tính số học không thể được biểu thị dưới dạng một số.

- Kiểu String: được sử dụng để biểu diễn dữ liệu dạng văn bản.

Kiểu dữ liệu đối tượng

Edison Technology Academy
Learning by Doing

Đối tượng (Object): tập hợp các thuộc tính (properties). Giá trị thuộc tính có thể là giá trị thuộc bất kỳ kiểu nào, bao gồm các đối tượng khác, cho phép xây dựng cấu trúc dữ liệu phức tạp.

- Date Object: được tích hợp sẵn trong JavaScript để xử lý ngày, giờ, thời gian.
- Mảng (Array): tập hợp nhiều phần tử, trong đó mỗi phần tử sẽ được đánh dấu vị trí bằng chỉ mục.
- Set: lưu trữ các giá trị duy nhất thuộc bất kỳ kiểu nào.

Xác định kiểu dữ liệu

Edison Technology Academy
Learning by Doing

Sử dụng toán tử typeof:

```
console.log(typeof 42); // Kết quả: "number"

console.log(typeof 'apple'); // Kết quả: "string"

console.log(typeof true); // Kết quả: "boolean"

console.log(typeof undeclaredVariable); // Kết quả: "undefined"
```

Edison Technology
Academy
Learning by Doing

Toán tử trong JavaScript

Các loại toán tử trong Javascript

Edison Technology Academy
Learning by Doing

Các loại toán tử trong Javascript bao gồm:

- Toán tử số học
- Toán tử gán
- Toán tử so sánh
- Toán tử logic

Toán tử số học

Edison Technology Academy
Learning by Doing

```
/** Toán tử số học
 +    ⟹ Cộng
 -    ⟹ Trừ
 *    ⟹ Nhân
 **   ⟹ Lũy thừa
 /    ⟹ Chia
 %    ⟹ Chia lấy dư
 ++   ⟹ Tăng 1
 --   ⟹ Giảm 1
 */
```

Toán tử ++, – với tiền tố và hậu tố

Edison Technology Academy
Learning by Doing

- Tiền tố

```
var a = 5;

++a;

console.log(a);
```

- Hậu tố

```
var a = 5;

a++;

console.log(a);
```

Toán tử ++, – với tiền tố và hậu tố

Edison Technology Academy
Learning by Doing

- Tiền tố:
  Ví dụ:
  Var a = 5;
  Var output = ++a;
  console.log(output)

```
/** Toán tử ++, -- với tiền tố và hậu tố
 * Tiền tố
 */

var a = 5;

var output = ++a;
/**
 * Việc 1: Thực hiện phép tính: a = a + 1
 * Việc 2: Trả về giá trị của a
 */

console.log(output);
```

Toán tử ++, – với tiền tố và hậu tố

Edison Technology Academy
Learning by Doing

- Hậu tố:
  Ví dụ:
  Var a = 5;
  Var output = a++;
  console.log(output)

```
/** Toán tử ++, -- với tiền tố và hậu tố
 * Hậu tố
 */

var a = 5;

var output = a++;
/**
 * Việc 1: Copy giá trị của a sang 1 biến mới (gọi là b)
 * Việc 1: Thực hiện phép tính: a = a + 1
 * Việc 2: Trả về giá trị của b
 */

console.log(output);
```

Toán tử gán

Edison Technology Academy
Learning by Doing

```
/** Toán tử gán
 Toán tử      Ví dụ      Tương đương
 =            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
 */
```

Toán tử so sánh

Edison Technology Academy
Learning by Doing

```
/** Toán tử so sánh
 Toán tử        Mô tả
 ==             Bằng về giá trị
 ===            Bằng về giá trị và kiểu dữ liệu
 !=             Khác
 >              Lớn hơn
 <              Nhỏ hơn
 >=             Lớn hơn hoặc bằng
 <=             Nhỏ hơn hoặc bằng
 */
```

Toán tử logical

Edison Technology Academy
Learning by Doing

```
/** Toán tử logic
 Toán tử        Mô tả
 &&             And
 ||             Or
 !              Not
 */
```

Edison Technology
Academy
Learning by Doing

Làm việc với chuỗi trong JavaScript

Chuỗi

Edison Technology Academy
Learning by Doing

- Chuỗi là một loại dữ liệu biểu diễn các ký tự.
- Các chuỗi được định nghĩa bằng cách sử dụng dấu ngoặc đơn hoặc dấu ngoặc kép, tùy thuộc vào việc sử dụng các ký tự trích dẫn bên trong chuỗi.

Các cách tạo chuỗi

Edison Technology Academy
Learning by Doing

```
/** Các cách tạo chuỗi
 * C1: Sử dụng dâu ngoặc đơn hoặc ngoặc kép
 */
var fullName = "Nguyễn Văn A";

// C2: Sử dụng đôi đượng String() trong Javascript
var name = new String("A");
```

Làm việc với chuỗi.

Edison Technology Academy
Learning by Doing

Một vài phương thức làm việc với chuỗi thường dùng:

- length: trả về độ dài của chuỗi.
- indexOf(): tìm kiếm một chuỗi con trong chuỗi và trả về vị trí xuất hiện đầu tiên của chuỗi con đó.
- slice(): trả về một phần của chuỗi, bắt đầu từ vị trí được chỉ định.
- replace(): thay thế một chuỗi con trong chuỗi bằng một chuỗi khác.
- toUpperCase(): chuyển đổi chuỗi thành chữ hoa.
- toLowerCase(): chuyển đổi chuỗi thành chữ thường.

Làm việc với chuỗi.

Edison Technology Academy
Learning by Doing

```
let myString = "Hello, World!";
console.log(myString.length); // 13
console.log(myString.indexOf("World")); // 7
console.log(myString.slice(0, 5)); // "Hello"
console.log(myString.replace("World", "JavaScript"));
// "Hello, JavaScript!"
console.log(myString.toUpperCase()); // "HELLO, WORLD!"
console.log(myString.toLowerCase()); // "hello, world!"
```

Edison Technology
Academy
Learning by Doing

Làm việc với số trong JavaScript

Số

- kiểu số (number) là một loại dữ liệu biểu diễn các giá trị số học. Kiểu số có thể biểu diễn các giá trị nguyên, thập phân, số âm, số dương, v.v.
- JavaScript hỗ trợ nhiều kiểu số khác nhau, bao gồm số nguyên (integer), số thực (floating-point), số thập lục phân (hexadecimal), số nhị phân (binary), v.v.

Số


```
/** Các cách tạo số
 * C1:
 */
var age = 25;

// C2: Sử dụng đối đượng Number() trong Javascript
var otherAge = new Number(25);
```

Làm việc với số

Edison Technology Academy
Learning by Doing

Một vài phương thức khi làm việc với số

- toFixed(): chuyển đổi số thành chuỗi với số lượng chữ số thập phân được xác định.
- toString(): chuyển đổi số thành chuỗi.
- parseInt(): chuyển đổi chuỗi thành số nguyên.
- parseFloat(): chuyển đổi chuỗi thành số thực.

Làm việc với số

Edison Technology Academy
Learning by Doing

```
let myNumber = 3.14159265359;
console.log(myNumber.toFixed(2)); // "3.14"
console.log(myNumber.toString()); // "3.14159265359"
console.log(parseInt("123")); // 123
console.log(parseFloat("3.14")); // 3.14
```

Edison Technology
Academy
Learning by Doing

Làm việc với mảng trong JavaScript

Mảng

Edison Technology Academy
Learning by Doing

- là một kiểu dữ liệu dùng để lưu trữ và quản lý tập hợp các giá trị có cùng kiểu dữ liệu.
- Mảng trong JavaScript có thể chứa các giá trị số, chuỗi, đối tượng và các giá trị khác.
- Các phần tử trong mảng được đánh chỉ số bắt đầu từ 0 cho đến n-1 (n là số lượng phần tử của mảng).

Cách khai báo và truy xuất giá trị

Edison Technology Academy
Learning by Doing

```
// Khai báo 1 mảng
let array = [1, 2, 3];

// Khai báo mảng bằng constructor
let array1 = new Array();

// Khai báo 1 mảng khi biết số lượng phần tử bằng constructor
let array2 = new Array(5);

// Khai báo 1 mảng khi biết giá trị phần tử bằng constructor
let array3 = new Array(1, 2, 3);

// Cách truy xuất giá trị
array3[0]; // 1
array3[1]; // 2
array3[2]; // 3
```

Các method khi làm việc với mảng

Edison Technology Academy
Learning by Doing

```
/** Các method thường dùng khi làm việc với mảng
 var array = [1,2,3]

 Tên method         Cách dùng             Mô tả
 push()             array.push(4)         Thêm 1 phần tử vào vị trí cuối cùng của mảng.
 pop()              array.pop()           Xóa phần tử cuối cùng của mảng và trả về phần đã xóa.
 shift()            array.shift()         Xóa phần tử đầu tiên của mảng và trả về phần tử đã xóa.
 unshift()          array.unshift(5,6)    Thêm 1 hoặc nhiều phần tử vào vị trí đầu tiên của mảng và trả về mảng mới.
 */
```

Các method khi làm việc với mảng

Edison Technology Academy
Learning by Doing

```
/** Một vài phương thức khác khi làm việc với mảng
 * Tên phương thức
 */
* toString()
* join()
* concat()
* splice()
* slice()
```

Các method khi làm việc với mảng

Edison Technology Academy
Learning by Doing

```
/** Một vài phương thức làm việc với mảng
*/
* forEach()
* map()
* filter()
* find()
* some()
* every()
* reduce()
```

Edison Technology
Academy
Learning by Doing

Làm việc với hàm trong JavaScript

Hàm

Edison Technology Academy
Learning by Doing

- Hàm trong JavaScript là một khối code được định nghĩa một lần, có thể được gọi lại nhiều lần để thực thi một tác vụ cụ thể
- Hàm thường được sử dụng để tái sử dụng code và giúp code của chương trình dễ đọc hơn.
- Một hàm bao gồm các thành phần sau:
- Tên hàm: Đây là tên của hàm, được sử dụng để gọi hàm.
- Tham số: Đây là các giá trị truyền vào hàm để xử lý.
- Câu lệnh: Đây là mã để thực thi một tác vụ cụ thể.
- Giá trị trả về: Đây là giá trị được trả về sau khi hàm được thực thi.

Các khai báo hàm

Edison Technology Academy
Learning by Doing

```
// Cách khai báo hàm
function getName(name) {
  return "Xin chào" + " " + name;
}

// Cách gọi hàm
getName("ABC");
```

Edison Technology
Academy
Learning by Doing

Callback function trong JavaScript

Callback function

Edison Technology Academy
Learning by Doing

- Callback là một hàm được truyền vào một hàm khác như một đối số.
- Hàm nhận Callback sẽ gọi lại (execute) hàm được truyền vào đó trong một thời điểm nào đó.

Callback function

Edison Technology Academy
Learning by Doing

```
function multiply(a, b, callback) {
  var result = a * b;
  callback(result);
}

function printResult(result) {
  console.log('Result is: ' + result);
}

multiply(5, 10, printResult); // Output: Result is: 50
```

Edison Technology
Academy
Learning by Doing

Làm việc với đối tượng trong JavaScript

Đối tượng

Edison Technology Academy
Learning by Doing

- Object là một kiểu dữ liệu phổ biến được sử dụng để lưu trữ và tổ chức dữ liệu trong chương trình.
- Một object bao gồm một tên (name) và một tập các thuộc tính (properties) và phương thức (methods).
- Thuộc tính (properties) của một đối tượng là các giá trị được lưu trữ trong đối tượng
- phương thức (methods) là các hành động mà đối tượng có thể thực hiện.

Làm việc với đối tượng

Edison Technology Academy
Learning by Doing

```
// Cách khai báo đối tượng
// C1
var myInfo = {
  name: "A",
  age: 20,
};

// C2
var myInfo2 = new Object();
myInfo2.name = "A";
myInfo2.age = 20;

// Cách lấy giá trị trong đối tượng
myInfo.name; // A
myInfo2["age"]; // 20
```

Object constructor

Edison Technology Academy
Learning by Doing

- Object Constructor là một hàm được sử dụng để tạo đối tượng mới
- Khi chúng ta tạo một đối tượng bằng Object Constructor, JavaScript sẽ tạo một thể hiện mới của đối tượng đó
- Object Constructor có thể được sử dụng để tạo đối tượng với các thuộc tính và phương thức khác nhau.

Cách khởi tạo Object Constructor

Edison Technology Academy
Learning by Doing

```
// Cách tạo object constructor
function ObjectName(property1, property2, ... , propertyN) {
  this.property1 = property1;
  this.property2 = property2;
  ...
  this.propertyN = propertyN;
  this.method1 = function() {
    // do something
  }
  this.method2 = function() {
    // do something
  }
  ...
  this.methodN = function() {
    // do something
  }
}

// Khởi tạo Object từ object constructor
var newObj = new ObjectName(value1, value2, ... , valueN);
```

Edison Technology
Academy
Learning by Doing

Cơ bản về prototype trong JavaScript

Khái niệm prototype

Edison Technology Academy
Learning by Doing

Prototype là cơ chế mà các đối tượng JavaScript kế thừa các tính năng từ một đối tượng khác.

Mọi đối tượng trong JavaScript đều có một thuộc tính được dựng sẵn là prototype, và chúng thừa kế thuộc tính cũng như phương thức từ prototype của mình.

Sử dụng thuộc tính prototype

Edison Technology Academy
Learning by Doing

Thuộc tính prototype cho phép bạn thêm các thuộc tính mới vào các hàm object constructor

```
function Person(first, last, age) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
}

// Thêm thuộc tính nationality vào object constructor
Person.prototype.nationality = "English";

// Tạo đối tượng
const person1 = new Person('John', 'Doe', 22);
const person2 = new Person('Sara', 'Kent', 20);

// Đối tượng person1 và person2 được kế thừa thuộc tính nationality từ prototype
console.log(person1.nationality);
console.log(person2.nationality);

// Kết quả
// English
// English
```

Edison Technology
Academy
Learning by Doing

Cấu trúc điều khiển trong JavaScript

Cấu trúc if...else

Edison Technology Academy
Learning by Doing

Câu lệnh if thực hiện một câu lệnh nếu một điều kiện được chỉ định là đúng (truthy). Nếu điều kiện là sai (falsy), một câu lệnh khác trong mệnh đề else tùy chọn sẽ được thực hiện.

Lưu ý: Trong JavaScript, giá trị truthy là giá trị được coi là true khi gặp trong ngữ cảnh Boolean. Tất cả các giá trị đều là truthy trừ khi chúng được định nghĩa là falsy:

- false, 0, -0, 0n, "", null, undefined và NaN

Cú pháp if...else

Edison Technology Academy
Learning by Doing

Cú pháp

```
/ Câu trúc với mệnh đề If
if (điều kiện) {
  câu lệnh 1
}

// Câu trúc với mệnh đề If-else
if (điều kiên) {
  câu lệnh 1
} else {
  câu lệnh 2
}

// Câu trúc với nhiêu mệnh đề If-else
if (điều kiên) {
  câu lệnh 1
} else if (điêu kiện 2) {
  câu lệnh 2
} else if (điêu kiện 3) {
  câu lệnh 3
}
//  ...
else {
  câu lệnh N
}
```

Ví dụ cú pháp if...else

Edison Technology Academy
Learning by Doing

```
function testNumber(a) {
  let result;
  if (a > 0) {
    result = 'là số dương';
  } else {
    result = 'không phải là số dương';
  }
  return result;
}
console.log(testNumber(10));

// Kết quả: "là số dương"
```

Edison Technology
Academy
Learning by Doing

Vòng lặp trong JavaScript

Cấu trúc switch

Edison Technology Academy
Learning by Doing

- Câu lệnh **switch** sẽ đánh giá một biểu thức, so khớp giá trị của biểu thức với một loạt mệnh đề case và thực hiện các câu lệnh sau mệnh đề case đầu tiên với một giá trị phù hợp, cho đến khi gặp câu lệnh **break**.
- Mệnh đề **default** của câu lệnh **switch** sẽ được thực thi nếu không có trường hợp nào phù hợp với giá trị của biểu thức.

Cú pháp

Edison Technology Academy
Learning by Doing

```
switch (biểu thức) {
  case value1:
    //Câu lệnh sẽ được thực thi khi kết quả của biểu thức khớp với value1
    break;
  case value2:
    //Câu lệnh sẽ được thực thi khi kết quả của biểu thức khớp với value2
    break;
  ...
  case valueN:
    //Câu lệnh sẽ được thực thi khi kết quả của biểu thức khớp với valueN
    break;
  default:
    //Câu lệnh sẽ được thực thi khi không có giá trị nào khớp với giá trị của biểu thức
    break;
}
```

Ví dụ cấu trúc switch

Edison Technology Academy
Learning by Doing

```
const expr = 'Papayas';
switch (expr) {
  case 'Oranges':
    console.log('Oranges are $0.59 a pound.' );
    break;
  case 'Mangoes':
  case 'Papayas':
    console.log('Mangoes and papayas are $2.79 a pound.' );
    break;
  default:
    console.log(`Sorry, we are out of ${expr}.`);
}
 // Kết quả: "Mangoes and papayas are $2.79 a pound."
```

Câu lệnh for

Edison Technology Academy
Learning by Doing

Câu lệnh **for** tạo một vòng lặp bao gồm ba biểu thức tùy chọn, được đặt trong dấu ngoặc đơn và được phân tách bằng dấu chấm phẩy, theo sau là một câu lệnh (thường là câu lệnh khối) sẽ được thực hiện.

Cú pháp

```
for (khởi tạo; điều kiện; bước nhảy)
  câu lệnh
```

Ví dụ

```
for (let i = 0; i < 9; i++) {
      console.log(i);
}
```

Câu lệnh while

Edison Technology Academy
Learning by Doing

Câu lệnh **while** tạo ra một vòng lặp thực hiện một câu lệnh được chỉ định miễn là điều kiện kiểm tra đánh giá là true. Điều kiện được đánh giá trước khi thực hiện câu lệnh.

Cú pháp

```
while (điều kiện){
  câu lệnh
}
```

Ví dụ

```
let n = 0;
while (n < 3) {
  n++;
}
console.log(n);
// Kết quả: 3
```

Câu lệnh do-while

Edison Technology Academy
Learning by Doing

Câu lệnh **do-while** tạo ra một vòng lặp thực hiện một câu lệnh được chỉ định cho đến khi điều kiện kiểm tra đánh giá là false. Điều kiện được đánh giá sau khi thực hiện câu lệnh, do đó câu lệnh được chỉ định được thực hiện ít nhất một lần.

Cú pháp

```
Do{
câu lệnh
} while (điều kiện)
```

Ví dụ

```
let result = '';let i = 0;

do {
  i = i + 1;

  result = result + i;
} while (i < 5);

console.log(result);

// Kết quả: "12345"
```

Từ khóa break và continue

Edison Technology Academy
Learning by Doing

break được sử dụng để thoát khỏi vòng lặp

Ví dụ

```
let i = 0;
while (i < 6) {

  if (i === 3) {
    break;
  }
  i = i + 1;
}
console.log(i);


// Kết quả: 3
```

continue chấm dứt thực thi các câu lệnh trong vòng lặp hiện tại và tiếp tục thực hiện vòng lặp với lần lặp tiếp theo.

Ví dụ

```
let text = '';
for (let i = 0; i < 10; i++) {
  if (i === 3) {
    continue;
  }
  text = text + i;
}
console.log(text);
 // Kết quả: "012456789"
```

Edison Technology
Academy
Learning by Doing

Xử lý ngoại lệ với cấu trúc try-catch

Cấu trúc try-catch

Edison Technology Academy
Learning by Doing

Bao gồm khối **try**, **catch**, **finally**. Code trong khối **try** được thực thi đầu tiên và nếu nó ném ra một lỗi, code trong khối **catch** sẽ được thực thi. Code trong khối **finally** sẽ luôn được thực thi trước khi luồng điều khiển thoát ra khỏi toàn bộ cấu trúc.

Cú pháp

```
try {
  // Khối mã (code block) chứa mã có thể gây ra lỗi
} catch (exception) {
  // Khối mã (code block) được thực thi khi xảy ra lỗi, tham số exception sẽ lưu trữ thông tin về lỗi đó
} finally {
  // Khối mã (code block) được thực thi sau khi try-catch đã kết thúc
}
```

Cấu trúc try-catch

Edison Technology Academy
Learning by Doing

Có 3 kiểu viết cấu trúc try:

- try...catch
- try...finally
- try...catch...finally

Ví dụ về try-catch

Edison Technology Academy
Learning by Doing

```
try {
  // Khối mã (code block) có thể gây ra lỗi
  var x = y + 10; // y chưa được định nghĩa
} catch (exception) {
  // Khối mã (code block) được thực thi khi có lỗi xảy ra
  console.log(exception.name + ': ' + exception.message);
} finally {

// Khối mã (code block) được thực thi sau khi try-catch đã kết
 thúc
  console.log('Try-catch block finished.');
}
```