# Bài 3

Hiểu rõ về câu lệnh if-else, switch

## [**Giá trị Truthy/Falsy**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648181#overview)

* **Giá trị Falsy**

- Có 5 giá trị Falsy (được coi là false) trong JS là: 0, undefined, null, NaN, " (empty string). Tất nhiên, false chính là false.

* **Giá trị Truthy**

 Trừ các giá trị Falsy thì các giá trị khác sẽ được chuyển đổi thành true

* **Toán tử ===  so sánh cân bằng nghiêm ngặt**

Toán tử === được gọi là toán tử bằng nghiêm ngặt vì nó không thực hiện ép kiểu, vì vậy nó chỉ trả về true khi cả 2 giá trị so sánh hoàn toàn giống nhau

* **Toán tử ==  so sánh cân bằng lỏng lẻo**

Toán tử == được gọi là toán tử bằng lỏng lẻo vì nó thực hiện ép kiểu trước khi so sánh. Ví dụ "18" == 18 // return true

* **Câu lệnh prompt**

Câu lệnh prompt được sử dụng để tạo ra một cửa sổ nhắc nhở, sẽ có một ô input để nhập

* **Cấu trúc *else if***

Cách sử dụng else if:

if (condition1) {

} else if (condition 2) {

} else {

}

* **Toán tử so sánh khác ( != )**

- Toán tử strict : !==

- Toán tử loose: !=

## [**Toán tử logic**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648191#overview)

* **Phép *AND - &&***

 Toán tử AND trả về true khi cả tất cả giá trị đều là true

* **Phép *OR - ||***

 Toán tử OR trả về true khi ít nhất một giá trị bằng true

* **Phép *NOT - !***

Toán tử NOT chỉ hoạt động trên 1 giá trị Boolean có vai trò đảo ngược giá trị của nó, từ true sang false và ngược lại

## [**Lệnh switch**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648197#overview)

Câu lệnh switch là một loại cơ chế điều khiển lựa chọn được sử dụng để cho phép giá trị của một biến hoặc biểu thức thay đổi luồng điều khiển thực hiện chương trình thông qua một nhánh nhiều đường.

Trong switch, nếu không có câu lệnh trường hợp nào khớp, các câu lệnh mặc định sẽ thực thi.

* **So sánh giữa if/else và switch**

Sự khác biệt giữa if else và switch là:

- if else khối thực thi dựa trên đánh giá của biểu thức trong câu lệnh if

- switch chọn các câu lệnh để thực thi tùy thuộc vào biến đơn được chuyền cho nó

Nên chuyển từ if/else sang switch khi các biểu thức trong câu lệnh if chỉ so sánh một biến với nhiều giá trị.

## [**Câu lệnh và biểu thức**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648199#overview)

* **Định nghĩa về biểu thức**

- Biểu thức là một đoạn code tạo ra một giá trị . Ví dụ: 3+4 là một biểu thức

* **Định nghĩa về câu lệnh**

- Câu lệnh là một đoạn code lớn hơn được thực hiện nhưng tự nó không tạo ra giá trị. Ví dụ: câu lệnh If/else, câu lệnh gán,...

- Câu lệnh như là các câu mô tả hành động mà chúng ta muốn chương trình thực hiện

## [**Toán tử điều kiện (3 ngôi)**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648205#overview)

* **Ví dụ về ? và :**

Toán tử điều kiện cho phép viết một phép so sánh giống như if/else nhưng toàn bộ trên một dòng.

Ví dụ: age >= 18 ? console.log('I like to drink wine ') : console.log('I like to drink water ');

* **Định nghĩa về toán tử điều kiện**

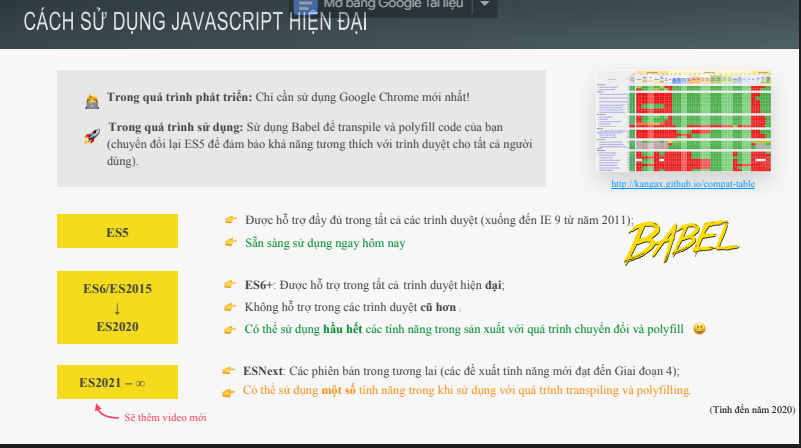
- Toán tử điều kiện còn được gọi là toán tử 3 ngôi gồm: điều kiện, phần hành động if, phần hành động else

- Toán tử điều kiện là một toán tử tạo ra giá trị hay toán tử là một biểu thức và có thể gán một biến bằng giá trị đó

Ví dụ : const drink = age >= 18 ? 'wine ' : 'water ';

# Lịch Sử





# [Bài 4 - Hàm và mảng](https://courses.funix.edu.vn/courses/course-v1:FUNiX+PRF192x_2.1-A_VN+2022_T3/course/#block-v1:FUNiX+PRF192x_2.1-A_VN+2022_T3+type@sequential+block@5969c8a0db7c4ff9be75e8aee8093100)

**Hàm (function)** hay chương trình con là một thuật ngữ chỉ một cụm câu lệnh được đặt tên để chương trình có thể gọi bất cứ lúc nào. Hàm là một khái niệm cực kì quan trọng trong lập trình, các bạn cần phải sử dụng hàm trong các bài lập trình nếu như không muốn phải copy và paste các đoạn code đơn giản khắp nơi trong chương trình của mình.

**Array (mảng)** là một loại dữ liệu dạng tập hợp nhiều phần tử, trong đó mỗi phần tử sẽ được đánh dấu vị trí bằng chỉ mục. Chúng ta thường sử dụng mảng để lưu trữ dữ liệu dạng danh sách. Ví dụ để lưu trữ danh sách tên sinh viên trong một lớp nào đó thì có thể sử dụng mảng, mỗi phần tử sẽ là tên của một thành viên trong lớp.

## [**Kích hoạt Strict Mode**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648215#overview)

* **Định nghĩa Strict Mode**

- Strict Mode là một chế độ đặc biệt mà chúng ta có thể kích hoạt trong JS để việc viết code dễ dàng và an toàn hơn, tránh các sai sót tình cờ

- Sử dụng 'use strict' ở đầu file code để kích hoạt

* **Ýnghĩa của việc sử dụng Strict Mode**

Lý do strict mode giúp tránh các sai sót trong code:

- Strict mode không cho phép dùng một số thứ như khai báo tên biến là một từ khóa trong JS, ...

- Strict mode sẽ tạo ra lỗi mà JS sẽ âm thầm bỏ qua, hiển thị lỗi đó cho chúng ta nhận ra được

Ví dụ: const if = true; // Uncaught SyntaxError: Unexpected token 'if'

## [**Hàm**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648219#overview)

* **Định nghĩa hàm**

- Hàm đơn giản là một đoạn code mà chúng ta có thể sử dụng lặp đi lặp lại trong code

- Hàm hơi giống biến nhưng nó là một tập các đoạn code, biến chứa giá trị còn hàm chưa một hoặc nhiều dòng code hoàn chỉnh

- Khai báo hàm bao gồm tên hàm, tham số truyền vào, thân hàm

## [**Arrow Function - Hàm mũi tên**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648227#overview)

Arrow Function hay hàm mũi tên là một biểu thức hàm nhưng được viết ngắn gọn hơn với dấu mũi tên ( => )

* + **Ưu điểm của arrow function (1:45)**

- Hàm mũi tên có cú pháp ngắn gọn, thường được sử dụng khi muốn viết một hàm trên một dòng đơn

* + **Arrow function với nhiều tham số đầu vào và nội dung hàm phức tạp (3:25)**

Hàm mũi tên cũng có thể truyền vào nhiều tham số và thân hàm nhiều hơn 1 dòng. Trường hợp hàm có nhiều hơn 1 tham số thì đặt chúng vào dấu ngoặc đơn. Trường hợp thân hàm có nhiều hơn 1 dòng cần có từ khóa return

## [**Hàm trong hàm**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648231#overview)

function cutFruitPieces(fruit) {

  return fruit \* 4;

}

function fruitProcessor(apples, oranges) {

  const applePieces = cutFruitPieces(apples);

  const orangePieces = cutFruitPieces(oranges);

  const juice = Juicewith${app≤Πeces}πeceofapp≤and${oran≥Πeces}πecesoforan≥.;

  return juice;

}

console.log(fruitProcessor(2, 3));

## [**Giới thiệu về array (mảng)**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648249#overview)

* + **Giới thiệu về array (0:00)**

- Array (mảng) là một cấu trúc dữ liệu quan trọng trong JS, cho phép lưu trữ các giá trị khác nhau vào trong một biến duy nhất

* + **Khai báo array (2:00)**

Có 2 cách khai báo mảng:

1. const array = [ item1, item2 ]

2. const array = new Array(item1, item2)

* + **Truy cập một phần tử trong array theo chỉ mục (4:40)**

- Chỉ số của Mảng bắt đầu từ 0 đến n-1 ( Mảng có n phần tử )

- Để truy cập đến phần tử thứ k thì ta gọi array[k-1]

## [**Các phương thức cơ bản của array**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648255#overview)

* + **push** – thêm một phần tử vào cuối array (0:50)

Ngoài việc thêm một phần tử vào array hàm push cũng trả về độ dài mới của array. Ví dụ:

const friends = ['Michael', 'Steven', 'Peter'];

// Add elements

const newLength = friends.push('Jay'); // 4

* + **unshift** – thêm một phần tử vào đầu array (3:55)

Ví dụ: const newLength = friends.unshift('John');

* + **pop** – gỡ một phần tử khỏi đuôi array (5:00)

Ví dụ: const popped = friends.pop();

* + **shift**– gỡ một phần tử khỏi đầu array (7:05)

Ví dụ: friends.shift(); // First

* + **indexof**- chỉ ra vị trí của 1 phần tử trong array (8:00)

Hàm index trả về vị trí của phần tử trong mảng và trả về -1 nếu không tìm thấy. Ví dụ:

const index = friends.indexOf('Steven') // 1

const index = friends.indexOf('Bob') // -1

* + **includes** – chỉ ra phần tử có xuất hiện trong array không (9:15)

Hàm includes trả về giá trị true/ false. Phương thức này sử dụng phép so sánh ===.  Ví dụ:

friends.includes('Steven') // true

friends.includes('Steven') // false

# Object

## [**Giới thiệu về Object**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648263#overview)

* + **Nhắc lại về array (0:00)**

- Với array, không có cách nào để đặt tên cho các phần tử, không thể tham chiếu các phần tử theo tên, mà chỉ có thể sử dụng số thứ tự của các phần tử đó.

* + **Giới thiệu về object với key-value (1:35)**

- Object quy định các phần tử theo cặp key-value

- Key giống như tên biến

- Value có thể thuộc bất kỳ kiểu dữ liệu nào, có thể gán value là một biểu thức. Mỗi cặp key-value trong object được gọi là thuộc tính của object

* + **Sự khác biệt giữa array và object (5:15)**

- Trong object lấy ra các giá trị dựa vào tên thuộc tính mà không cần quan tâm đến thứ tự của chúng, object sử dụng cho các dữ liệu phi cấu trúc

- Trong array thứ tự vô cùng quan trọng vì đó là cách truy cập vào phần tử, array sử dụng cho dữ liệu có thứ tự

## [**Dấu chấm và dấu ngoặc vuông với Object**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648271#overview)

* + **2 cách lấy giá trị của thuộc tính: dấu chấm và ngoặc vuông (1:05)**

Ví dụ:

Sử dụng dấu chấm: jonas.lastName

Sử dụng dấu ngoặc vuông: jonas['lastName']

* + **Khi nào thì chúng ta nên sử dụng cách nào (5:20)**

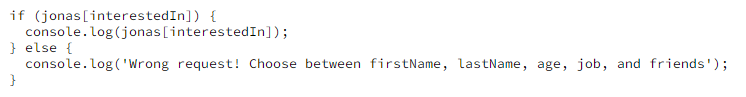
- Khi sử dụng ngoặc vuông ta có thể đặt bất kỳ biểu thức nào bên trong nó, linh hoạt trong một số trường hợp.

- Khi sử dụng dấu chấm ta phải biết chính xác tên thuộc tính.

* + **Cách xử lý khi thuộc tính đưa vào lấy không có trong object (9:20)**

Khi thuộc tính không có trong object thì sẽ trả về undefined, mà undefined là một giá trị false nên để chương trình không trả về lỗi không mong muốn thì nên check giá trị với câu lệnh if trước khi sử dụng giá trị đó.

Ví dụ:



* + Đưa thêm thuộc tính vào object (11:45)

Có thể sử dụng dấu chấm hoặc dấu ngoặc vuông để chỉ định thêm một thuộc tính cho object, đơn giản là gán giá trị cho thuộc tính đó.

Ví dụ:

https://firebasestorage.googleapis.com/v0/b/funix-way.appspot.com/o/xSeries%2FPRF192x%2FContent_Image%2FPRF192x_L5_H%C3%ACnh%204.PNG?alt=media&token=bdc0ebcb-8842-4646-982b-b36d506a901f

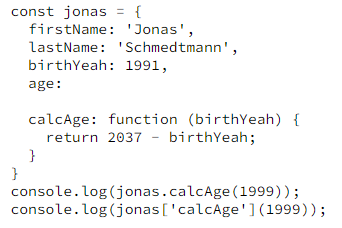
## [**Các phương thức trong Object**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648275#overview)

* + **Đưa hàm vào object – phương thức (method) (1:20)**

- Các hàm (function) thực sự chỉ là một kiểu giá trị, vì vậy chúng ta có thể tạo object với các khóa và giá trị là một hàm. Nói cách khác chúng ta có thể thêm phương thức vào đối tượng

- Gán giá trị cho một phương thức ta sử dụng biểu thức hàm (không thể dùng khai báo hàm)

Ví dụ:



* + Cú pháp **this**trong object (6:00)

- Để truy cập vào giá trị của một thuộc tính ngay trong object thì sử dụng tử khóa this

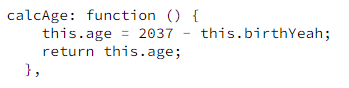
- Từ khóa this hay biến this về cơ bản là object (đối tượng) hiện tại của phương thức được gọi

* + Tại sao lại cần **this**để trỏ vào object?(10.20)

- Cần this trỏ vào object để lấy ra được những thuộc tính cần để tính toán mà không cần phải truyền biến từ bên ngoài vào

* + Tạo 1 thuộc tính ngay trong phương thức (11:45)

- Tạo một thuộc tính trong phương thức ta dùng từ khóa this.ten\_thuoc\_tinh và gán giá trị vào cho nó. Ví dụ:



* + Các phương thức built-in (21:25)

- Các array là một dạng object đặc biệt, vì chúng cũng có các phương thức mà chúng ta có thể sử dụng để thao tác (pop, push, ...)

- Các phương thức built-in là các phương thức được dựng sẵn để thao tác, về cơ bản nó cũng là các phương thức của object

# Vòng lặp

## [**Vòng lặp for**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648285#overview)

* + Định nghĩa về vòng lặp (0:00)

- Vòng lặp là một cấu trúc điều khiển

- Vòng lặp cho phép tự động hóa các tác vụ lặp đi lặp lại

* + Cấu trúc vòng lặp for và ví dụ (2:25)

- Vòng lặp bắt đầu với từ khóa for, biểu thức vòng lặp và phần thân

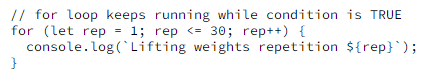
- Biểu thức vòng lặp gốm 3 phần, cách nhau bởi dấu chấm phẩy:

1. Giá trị ban đầu của bộ đếm

2. Điều kiện logic được đánh giá trước mỗi lần lặp, nếu điều kiện là true thì vòng lặp sẽ tiếp tục, nếu điều kiện là false thì vòng lặp sẽ dừng lại

3. Tăng bộ đếm

Ví dụ:

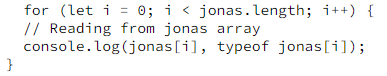


## [**Lặp trong array, break và continue**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648291#overview)

* + Sử dụng vòng lặp for để duyệt các phần tử của array (1:00)
  + Sử dụng vòng lặp for để duyệt qua các phần tử của mảng (array)

- Thứ tự của các phần tử trong array bắt đầu từ 0 nên khi duyệt bằng vòng lặp for, bộ đếm nên bắt đầu từ 0

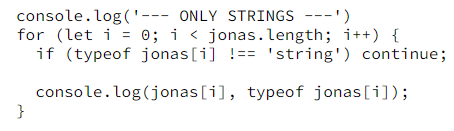
Ví dụ sử dụng for để duyệt qua các phần tử trong mảng:



* + Kết hợp vòng lặp và array để thực hiện các toán tử với array (12:20)
  + Câu lệnh **continue** (16:40)

- Continue là việc bỏ qua việc lặp lại hiện tại của vòng lặp mà thực hiện lần lặp tiếp theo.

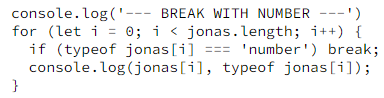
Ví dụ:



* + Câu lệnh **break**(19:55)

- Break được sử dụng để dừng lại hoàn toàn vòng lặp

Ví dụ:



## [**Vòng lặp while**](https://funix.udemy.com/course/the-complete-javascript-course/learn/lecture/22648303#overview)

* + So sánh vòng lặp while và for (1:10)

- Vòng lặp while cũng có các phần giống vòng lặp for, vẫn có biến đếm, một điều kiện để so sánh và tăng / giảm biến đếm bằng cách nào đó.

- Tuy nhiên vòng lặp while khác với for là ở biểu thức vòng lặp chỉ cần điều kiện lặp.

Ví dụ:

