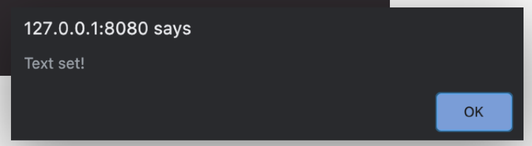
# CODE KHÔNG ĐỒNG BỘ

BLOCKING

```
const p = document.querySelector('.p');
p.textContent = 'My name is Jonas!';
alert('Text set!');
p.style.color = 'red';
```





Một phần của bối cảnh thực thi là thực thi code trong CPU của máy tính

## ĐỒNG BỘ

- 👉 Hầu hết các code là đồng bộ;
- 👉 Code đồng bộ được thực hiện theo dòng;
- 👉 Mỗi dòng code phải chờ dòng trước đó kết thúc;
- 👎 Việc chạy một hoạt động dài **chặn** việc thực thi code.

# CODE KHÔNG ĐỒNG BỘ

Không đồng bộ

CALLBACK SẼ CHẠY SAU TIMER

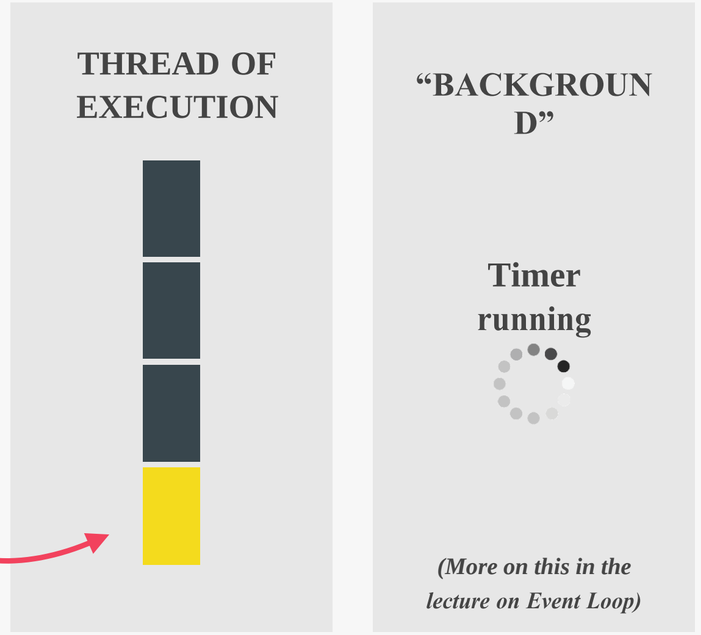
```
const p = document.querySelector('.p');
setTimeout(function () {
  p.textContent = 'My name is Jonas!';
}, 5000);
p.style.color = 'red';
```

👉 **Ví dụ:** Timer với callback

```
[1, 2, 3].map(v ⇒ v * 2);
```

Callback KHÔNG tự động tạo code không đồng bộ!

Được thực thi sau tất cả các code khác




## Không đồng bộ

👉 Code không đồng bộ được thực thi sau một tác vụ kết thúc

👍 Code không đồng bộ là **non-blocking**;

👉 Việc thực thi không chờ đợI một nhiệm vụ không đồng bộ để hoàn thành công việc.

👉 Các hàm callback một mình KHÔNG làm cho code không đồng bộ!

Điều phối hành vi của một chương trình trong một khoảng thời gian

# CODE KHÔNG ĐỒNG BỘ

Không đồng bộ

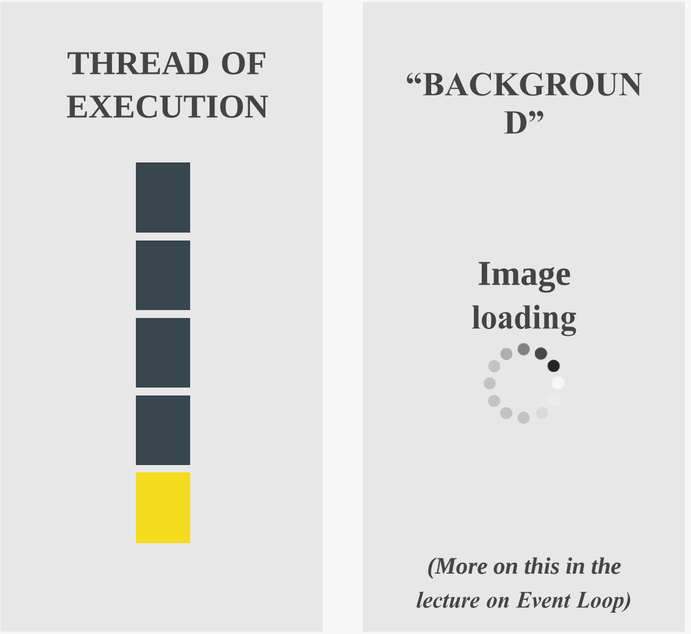
```
const img = document.querySelector('.dog');
img.src = 'dog.jpg';
img.addEventListener('load', function () {
  img.classList.add('fadeIn');
});
p.style.width = '300px';
```

CALLBACK SẼ CHẠY SAU SỰ KIỆN LOADS

👉 **Ví dụ:** Tải hình ảnh không đồng bộ với sự kiện và callback

👉 **Các ví dụ khác**: API định vị địa lý hoặc cuộc gọi AJAX

`addEventListener` KHÔNG tự động tạo code không đồng bộ!




## KHÔNG ĐỒNG BỘ

Điều phối hành vi của một chương trình trong một khoảng thời gian

👉 Code không đồng bộ được thực thi sau một tác vụ trên background kết thúc;

👍 Code không đồng bộ là **non-blocking**;

👉 Việc thực thi không chờ đợI một nhiệm vụ không đồng bộ để hoàn thành công việc;

👉 Các hàm callback một mình KHÔNG làm cho code không đồng bộ!

# AJAX CALLS LÀ GÌ?

## AJAX

**A**synchronous **J**avaScript **A**nd **X**ML: Cho phép chúng ta giao tiếp với các máy chủ web từ xa một cách không đồng bộ. Với các lệnh gọi AJAX, chúng ta có thể yêu cầu dữ liệu từ các máy chủ web động.

# API LÀ GÌ?

## API

- **A**pplication **P**rogramming **I**nterface: Phần mềm có thể được sử dụng bởi một phần mềm khác, cho phép các ứng dụng nói chuyện với nhau;

- Có nhiều loại API trong lập trình web:
  - DOM API
  - Geolocation API
  - Own Class API
  - **"Online" API**

Just "API"

- "API trực tuyến": Ứng dụng chạy trên máy chủ, nhận yêu cầu dữ liệu và gửi dữ liệu trở lại dưới dạng phản hồi;

- Chúng ta có thể xây dựng API của riêng mình (yêu cầu lập trình back-end với node.js) hoặc sử dụng API của bên thứ 3

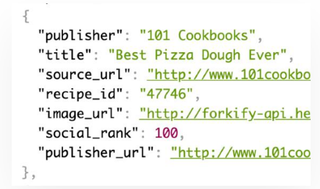

There is an API for *everything*

- Dữ liệu thời tiết
- Dữ liệu về các quốc gia
- Dữ liệu chuyến bay
- Dữ liệu chuyển đổi tiền tệ
- API gửi email hoặc SMS
- Google Maps
- *Rất nhiều khả năng...*

## AJAX

**Định dạng dữ liệu XML**

**Định dạng dữ liệu JSON**




Định dạng dữ liệu API phổ biến nhất

# PROMISE LÀ GÌ?

## PROMISE

👉 **Promise:** Object được sử dụng như placehoder cho kết quả tương lai của một hoạt động không đồng bộ.

↓ Less formal

👉 **Promise:** Container cho một giá trị được phân phối không đồng bộ.

↓ Less formal

👉 **Promise:** Container cho một giá trị tương lai.

↖ **Ví dụ:** Phản hồI từ lệnh gọi AJAX

👉 Chúng ta không còn phải dựa vào các event và callback được truyền vào các hàm không đồng bộ để xử lý kết quả không đồng bộ;

👉 Thay vì lồng các callback, chúng ta có thể **nối chuỗi promise** cho một chuỗi hoạt động không đồng bộ: **thoát khỏi callback hell** 🎉


**Promise:** tôi sẽ nhận được tiền nếu tôi đoán chính xác kết quả

🤞 Tôi mua vé số (promise) ngay bây giờ

↓

🔮 Rút thăm xổ số diễn ra không đồng bộ

↓

💰 Nếu kết quả đúng, tôi sẽ nhận được tiền, vì điều này đã được cam kết

# THE PROMISE LIFECYCLE

**Trước khi** giá trị tương lai được xác định

Nhiệm vụ không đồng bộ **đã kết thúc**

**PENDING**



ASYNC TASK

**SETTLED**

**FULFILLED**

Thành công! Giá trị **được xác định**

**REJECTED**

Xảy ra lỗi

👉 Chúng ta có thể **xử lý** các trạng thái khác nhau trong code!

Ví dụ: Fetch API trả về promise

XÂY DỰNG PROMISE

SỬ DỤNG PROMISE

Khi chúng ta đã có một promise, chẳng hạn: promise được trả về từ Fetch API