

# LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 1: GIỚI THIỆU VỀ LẬP TRÌNH ĐA NỀN  
TẢNG (REACT NATIVE)

PHẦN 1: GIỚI THIỆU REACT NATIVE

- ☐ Giới thiệu chung về React Native
- ☐ Phạm vi ứng dụng của React Native
- ☐ Xu hướng phát triển React Native
- ☐ Thiết lập môi trường lập trình cho Window và MacOS
- ☐ Ôn tập lại kiến thức javascript cơ bản

- ☐ React Native là một framework mã nguồn mở cho phép các nhà phát triển xây dựng ứng dụng di động dựa trên JavaScript và thư viện React. Nó cho phép xây dựng ứng dụng di động cho cả iOS và Android bằng cách sử dụng các thành phần native của thiết bị.
- ☐ React Native giúp tăng tốc độ phát triển và cung cấp trải nghiệm người dùng tốt hơn so với các framework khác. Nó cũng có một cộng đồng lớn với nhiều thư viện hữu ích để giúp phát triển ứng dụng di động một cách nhanh chóng và dễ dàng hơn.

## Các tính năng của React Native

---

- ☐ **Cross-platform:** React Native cho phép bạn xây dựng ứng dụng cho cả iOS và Android, từ cùng một codebase.
- ☐ **Hot Reloading:** Hot Reloading là một tính năng của React Native cho phép bạn xem kết quả của các thay đổi code của bạn trực tiếp trên thiết bị hoặc giả lập mà không cần phải khởi động lại ứng dụng.
- ☐ **Thư viện đa dạng:** React Native có rất nhiều thư viện hữu ích để giúp bạn xây dựng các tính năng của ứng dụng như định vị, mạng, và các thành phần giao diện.
- ☐ **Cộng đồng lớn:** React Native có cộng đồng lớn với nhiều tài liệu hướng dẫn, ví dụ code, và các nguồn tài nguyên khác để giúp bạn giải quyết các vấn đề mà bạn gặp phải trong quá trình phát triển ứng dụng của mình.

## Ưu điểm của React Native

---

- ☐ **Tiết kiệm thời gian và chi phí:** Vì React Native cho phép xây dựng ứng dụng cho cả iOS và Android từ cùng một codebase, nên việc phát triển ứng dụng sẽ nhanh hơn và tiết kiệm chi phí hơn so với phát triển các ứng dụng trên các nền tảng riêng biệt.
- ☐ **Codebase đơn giản và dễ dàng bảo trì:** Vì React Native sử dụng JavaScript để xây dựng ứng dụng, nên codebase của nó rất dễ đọc và bảo trì. Ngoài ra, các thành phần của ứng dụng được tách biệt rõ ràng, giúp cho việc thay đổi, bảo trì và kiểm tra chức năng dễ dàng hơn.
- ☐ **Hiệu suất cao:** React Native sử dụng kiến trúc khác biệt so với các công nghệ khác để tạo ra các thành phần giao diện, đảm bảo hiệu suất cao hơn.

## Nhược điểm của React Native

---

- ☐ **Thời gian khởi động ban đầu:** Do việc tải và khởi động một số thành phần của ứng dụng, việc khởi động ứng dụng React Native có thể lâu hơn so với một ứng dụng native.
- ☐ **Giới hạn chức năng của các thành phần:** Các thành phần được cung cấp bởi React Native có thể giới hạn chức năng so với các thành phần tương đương trong các ứng dụng native.
- ☐ **Sự phụ thuộc vào bên thứ ba:** Việc sử dụng các thư viện bên thứ ba có thể dẫn đến sự phụ thuộc vào các bên thứ ba không được kiểm soát và có thể gây ra các vấn đề về bảo mật hoặc tính ổn định.
- ☐ **Cập nhật phiên bản:** Việc cập nhật phiên bản của React Native có thể gây ra một số vấn đề không tương thích với các thành phần khác trong ứng dụng.

## Nhược điểm của React Native

---

- ☐ **Thời gian khởi động ban đầu:** Do việc tải và khởi động một số thành phần của ứng dụng, việc khởi động ứng dụng React Native có thể lâu hơn so với một ứng dụng native.
- ☐ **Giới hạn chức năng của các thành phần:** Các thành phần được cung cấp bởi React Native có thể giới hạn chức năng so với các thành phần tương đương trong các ứng dụng native.
- ☐ **Sự phụ thuộc vào bên thứ ba:** Việc sử dụng các thư viện bên thứ ba có thể dẫn đến sự phụ thuộc vào các bên thứ ba không được kiểm soát và có thể gây ra các vấn đề về bảo mật hoặc tính ổn định.
- ☐ **Cập nhật phiên bản:** Việc cập nhật phiên bản của React Native có thể gây ra một số vấn đề không tương thích với các thành phần khác trong ứng dụng.

## Cài đặt môi trường Android cho Window

Lưu ý: những hướng dẫn ở dưới đây có thể sẽ không tương thích với các phiên bản mới của React Native. Vậy nên để cập nhật cách cài đặt môi trường phiên bản mới nhất, bạn vui lòng truy cập trang.

<https://reactnative.dev/docs/environment-setup?guide=native&platform=android&os=windows>

Lần cập nhật tài liệu gần nhất: 04/2023






## □ Cài đặt NodeJS

❖ Truy cập trang: <https://nodejs.org/en/download>

**Downloads**

Latest LTS Version: **18.15.0** (includes npm 9.5.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

| LTS<br>Recommended For Most Users   | Current<br>Latest Features  |  |
|---|---|--|
| <br><b>Windows Installer</b><br><small>node-v18.15.0-x86.msi</small> | <br><b>macOS Installer</b><br><small>node-v18.15.0.pkg</small> | <br><b>Source Code</b><br><small>node-v18.15.0.tar.gz</small> |
| Windows Installer (.msi)  | 32-bit  | 64-bit   |
| Windows Binary (.zip)   | 32-bit  | 64-bit   |
| macOS Installer (.pkg)  | 64-bit / ARM64  |  |
| macOS Binary (.tar.gz)  | 64-bit  | ARM64  |
| Linux Binaries (x64)  | 64-bit  |  |
| Linux Binaries (ARM)  | ARMv7   | ARMv8  |
| Source Code   | node-v18.15.0.tar.gz  |  |

<https://nodejs.org/dist/v18.15.0/node-v18.15.0-x86.msi> **Platforms**

Nhấn chọn **Windows Installer**. Sau đó vào cài đặt file đã tải như bình thường

## Cài đặt Java JDK



### Truy cập trang:

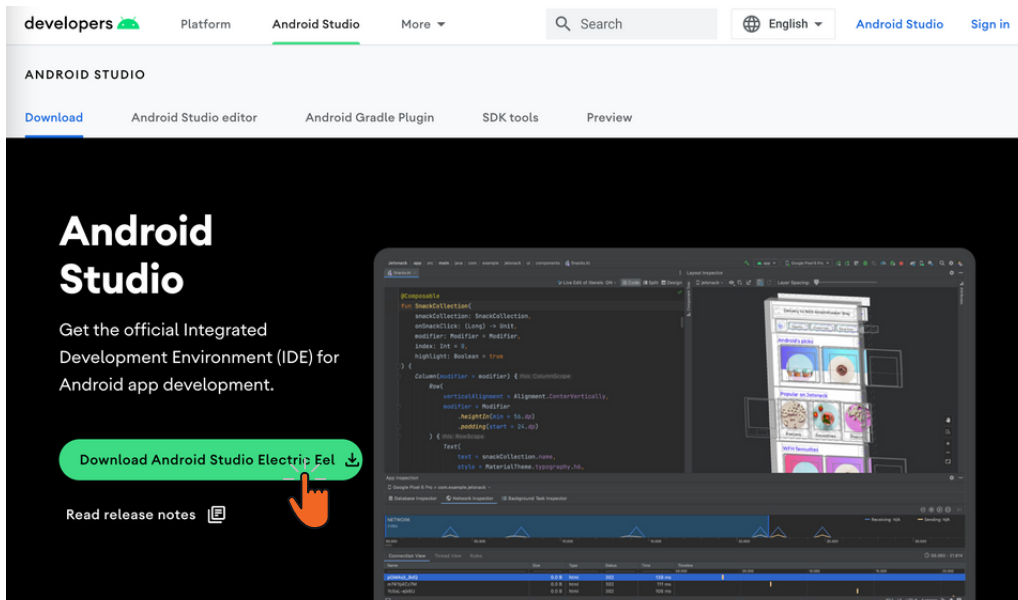
<https://www.oracle.com/in/java/technologies/javase/jdk11-archive-downloads.html>

| <b>Java SE Development Kit 11.0.17</b><br><small>This software is licensed under the <a href="#">Oracle Technology Network License Agreement for Oracle Java SE</a></small><br><small>JDK 11.0.17 <a href="#">checksum</a></small> |           |   |
|--|-----------|---|
| Product / File Description   | File Size | Download  |
| Windows x64 Installer  | 140.79 MB | <a href="#">jdk-11.0.17_windows-x64_bin.exe</a>  |
| Windows x64 Compressed Archive   | 158.43 MB | <a href="#">jdk-11.0.17_windows-x64_bin.zip</a>   |

Nhấn chọn **jdk-11.0.17\_windows-x64\_bin.exe**.  
 Sau đó vào cài đặt file đã tải như bình thường

## ❑ Cài đặt Android Studio

✦ **Truy cập trang:** <https://developer.android.com/studio>



Nhấn chọn **Download Android Studio Electric Eel**

## Cài đặt Android Studio trên Window

Lưu ý: những hướng dẫn ở dưới đây có thể sẽ không tương thích với các phiên bản mới của Android Studio trên Window. Vậy nên để cập nhật cách cài đặt Android Studio vào Android phiên bản mới nhất, bạn vui lòng truy cập trang.

<https://developer.android.com/studio/install>

## Cấu hình máy yêu cầu:

| Requirement       | Minimum   | Recommended                          |
|-------------------|---|--------------------------------------|
| OS                | 64-bit Microsoft Windows 8  | Latest 64-bit version of Windows     |
| RAM               | 8 GB RAM  | 16 GB RAM or more                    |
| CPU               | x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows <a href="#">Hypervisor Framework</a> . | Latest Intel Core processor          |
| Disk space        | 8 GB (IDE and Android SDK and Emulator)   | Solid state drive with 16 GB or more |
| Screen resolution | 1280 x 800  | 1920 x 1080                          |

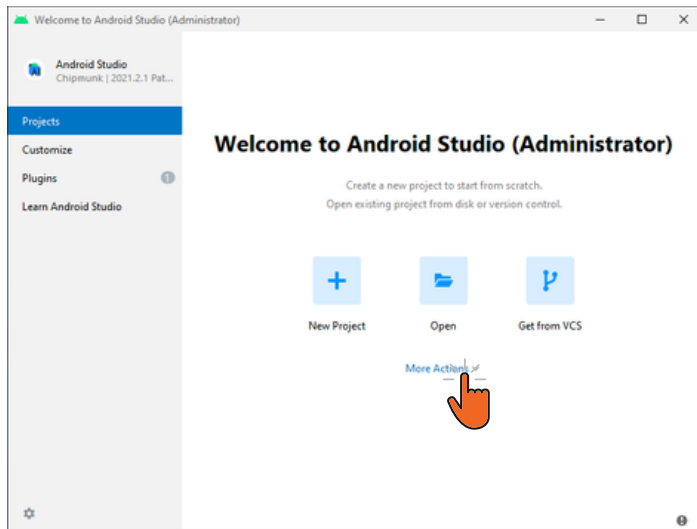
## ❖ Cài đặt Android Studio vào Window.

Để cài đặt Android Studio trên Window, làm theo hướng dẫn dưới đây:

- Nếu bạn cài đặt file .exe (khuyến khích), nhấn chuột phải để cài đặt nó.
- Nếu bạn cài đặt file .zip:
  - a. Giải nén file .zip
  - b. Copy thư mục android-studio vào thư mục **Program Files**.
  - c. Mở thư mục **android-studio > bin**
  - d. Nhấn chuột trái để cài đặt studio64.exe (cho máy 64-bit) hoặc studio.exe (cho máy 32-bit).
  - e. Sau đó, ứng dụng Android Studio sẽ mở lên, và bạn nhấn tùy chọn cài đặt theo mặc định.

## ❖ Cài đặt môi trường trong Android Studio

Sau khi cài đặt Android Studio thành công, cửa sổ như này sẽ hiện lên. Nhấn vào **More Actions**



Bạn có thể cài đặt SDK Manager trong Android Studio "**Preferences**", dưới cửa sổ **Appearance & Behavior** → **System Settings** → **Android SDK**.

Hãy chọn tab **"SDK Platforms"** từ trong **SDK Manager**, sau đó đánh dấu vào ô bên cạnh **"Show Package Details"** ở góc dưới bên phải. Tìm và mở rộng mục Android 13 (Tiramisu), sau đó đảm bảo rằng các mục sau được đánh dấu:

## **Android SDK Platform 33**

**Intel x86 Atom\_64 System Image** hoặc **Google APIs Intel x86 Atom System Image**

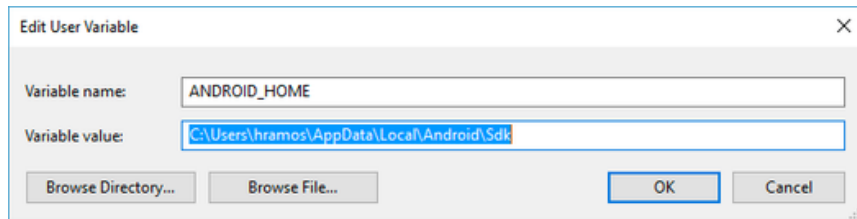
Tiếp theo, chọn tab **"SDK Tools"** và đánh dấu vào ô bên cạnh **"Show Package Details"** ở đây. Tìm và mở rộng mục **Android SDK Build-Tools**, sau đó đảm bảo rằng phiên bản **33.0.0** được chọn.

Cuối cùng, nhấn **"Apply"** để tải xuống và cài đặt Android SDK và các công cụ xây dựng liên quan.

## ❖ Cấu hình biến môi trường ANDROID\_HOME

Các công cụ React Native yêu cầu một số biến môi trường được thiết lập để xây dựng ứng dụng với mã nguồn native:

- Mở Windows **Control Panel**.
- Nhấp vào **User Account**, sau đó nhấp lại vào **User Account**
- Nhấp vào **Change my environment variables**
- Nhấp vào New ... để tạo một biến người dùng **ANDROID\_HOME** mới trỏ đến đường dẫn tới **Android SDK** của bạn:





SDK được cài đặt mặc định theo đường dẫn dưới đây:

```
%LOCALAPPDATA%\Android\Sdk
```

Bạn có thể tìm vị trí thực tế của SDK trong hộp thoại "**Settings**" của Android Studio, trong **Appearance & Behavior** → **System Settings** → **Android SDK**.

Mở một cửa sổ **Command Prompt** mới để đảm bảo rằng biến môi trường mới được tải trước khi tiến hành bước tiếp theo.

1. Mở **powershell**
2. Sao chép và dán **Get-ChildItem -Path Env:\** vào **powershell**
3. Xác minh rằng **ANDROID\_HOME** đã được thêm vào.

## ❖ Thêm đường dẫn platform-tools

1. Mở **Windows Control Panel**.
2. Nhấp vào **User Accounts**, sau đó nhấp lại vào **User Accounts**
3. Nhấp vào **Change my environment variables**
4. Chọn biến **Path**
5. Nhấp vào **Edit**
6. Nhấp vào **New** và thêm đường dẫn tới **platform-tools** vào danh sách.
7. Vị trí mặc định của thư mục này là:

**%LOCALAPPDATA%\Android\Sdk\platform-tools**

## Cài đặt môi trường React Native cho MacOS

Để cài đặt môi trường **android** và **ios** cho React Native trên **MacOS**. Bạn có thể vào link sau để xem chi tiết cách cài đặt môi trường.

Chi tiết hướng dẫn:

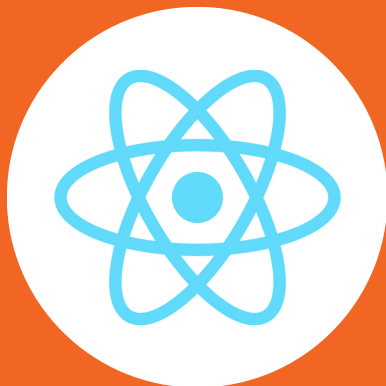
<https://reactnative.dev/docs/environment-setup?os=macos&platform=android>

## Cài đặt môi trường React Native cho Linux

Để cài đặt môi trường **android** cho React Native trên **Linux**. Bạn có thể vào link sau để xem chi tiết cách cài đặt môi trường.

Chi tiết hướng dẫn:

<https://reactnative.dev/docs/environment-setup?os=linux&platform=android>



# LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 1: GIỚI THIỆU VỀ LẬP TRÌNH ĐA  
NỀN TẢNG (REACT NATIVE)

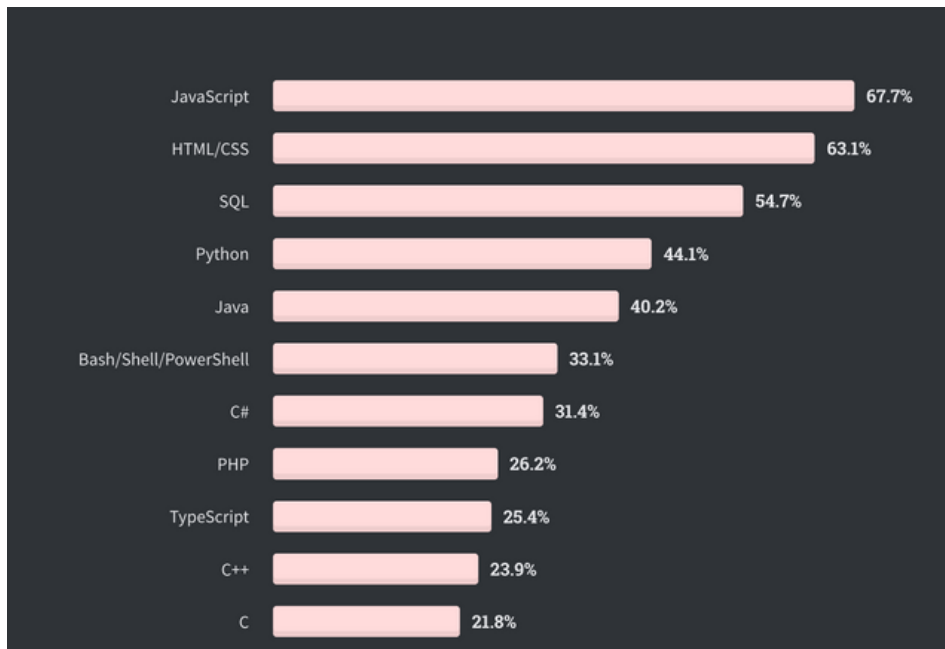
PHẦN 2: ÔN TẬP JAVASCRIPT CƠ BẢN

## ❖ Mục tiêu của bài học này

Để bắt đầu viết ứng dụng đa nền tảng bằng React Native, điều đầu tiên các bạn phải biết nền các kiến thức cơ bản của **Javascript**. Các bạn cũng đã được học một số kiến thức cơ bản từ những môn trước đó. Nhưng những kiến thức đó là chưa đủ, ở bài học này sẽ giúp bạn ôn lại và mở rộng thêm kiến thức về Javascript. Nó sẽ giúp ích rất lớn cho chặng đường lập trình ứng dụng di động đa nền tảng của bạn.

## ❖ Ngôn ngữ Javascript là gì?

Hiện tại ở thời điểm viết slide này, **Javascript** là ngôn ngữ có cộng đồng lập trình viên sử dụng nó phổ biến nhất, chiếm **67.7%** được thống kê bởi trang **Stackoverflow**



Các ngôn ngữ lập trình phổ biến năm 2023 (nguồn: Stack Overflow)

**JavaScript** là ngôn ngữ kịch bản đa nền tảng, hướng đối tượng được sử dụng để làm cho các trang web tương tác (ví dụ: có hoạt ảnh phức tạp, nút có thể nhấp, menu bật lên, v.v.). Ngoài ra còn có các phiên bản **JavaScript** phía máy chủ nâng cao hơn như **Node.js**, cho phép bạn thêm nhiều chức năng hơn vào trang web hơn là tải xuống tệp (chẳng hạn như cộng tác thời gian thực giữa nhiều máy tính). Bên trong môi trường máy chủ (ví dụ: trình duyệt web), **JavaScript** có thể được kết nối với các đối tượng của môi trường của nó để cung cấp quyền kiểm soát theo chương trình đối với chúng.

Ngôn ngữ **JavaScript** hiện tại đã quá phổ biến, và mạnh mẽ tới mức đã được áp dụng vào hầu hết các lĩnh vực công nghệ thông tin:

- Phát triển web
- Phát triển ứng dụng di động
- Phát triển máy chủ
- Phát triển game
- Phát triển trí tuệ nhân tạo

## ❖ Khai báo biến trong javascript

Ở **Javascript** có 3 cách khai báo biến:

- **var** - Khai báo một biến, có thể thay đổi giá trị của biến.
- **let** - Khai báo một biến cục bộ có thể thay đổi giá trị của biến.
- **const** - Khai báo một biến cục bộ, là biến chỉ đọc

Trong quá trình phát triển ứng dụng **React native** các bạn sẽ hầu như không thấy các biến được khai báo bằng **var**. Tại sao vậy? Bởi vì nó có rủi ro tiềm ẩn, hãy xem ví dụ bên dưới.

```
x = 5; // Assign 5 to x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x; // Display x in the element

var x; // Declare x
```

Hãy nhìn đoạn code bên trên và xem có gì lạ!



Đoạn code bên trên đã mắc một lỗi, thuật ngữ được gọi là **Hoisted**. Nó xảy ra khi bạn gán giá trị mới cho biến trước khi nó được khởi tạo. Ở ví dụ trên thì nó vẫn hoạt động bình thường, nhưng ở các ứng dụng lớn, phức tạp hơn thì các biến có thể sẽ xuất ra kết quả không mong muốn, các bạn cũng sẽ rất khó bảo trì, nâng cấp sau này.

Để tránh trường hợp **Hoisted** này xảy ra, **let** đã ra đời khiến cách khai báo biến **var** chỉ còn là kỉ niệm.

## ❖ Kiểu dữ liệu trong javascript

JavaScript cung cấp các kiểu dữ liệu khác nhau để chứa các loại giá trị khác nhau. Có hai loại kiểu dữ liệu trong JavaScript.

1. Kiểu dữ liệu primitive
2. Kiểu dữ liệu non-primitive (tham chiếu)

**JavaScript** là một ngôn ngữ kiểu động, có nghĩa là bạn không cần chỉ định loại biến vì nó được sử dụng linh động bởi công cụ JavaScript. Bạn cần sử dụng **var**, **let** hoặc **const** ở đây để chỉ định kiểu dữ liệu. Nó có thể chứa bất kỳ loại giá trị nào như number, string, v.v. Chẳng hạn:

```
let a=40; //holding number  
let b="Rahul"; //holding string
```

## ❖ Kiểu dữ liệu primitive trong javascript

Có năm loại kiểu dữ liệu nguyên thủy trong **JavaScript**. Gồm:

| Kiểu dữ liệu     | Mô tả  |
|------------------|--|
| <b>String</b>    | đại diện cho chuỗi ký tự, ví dụ: "Xin chào"      |
| <b>Number</b>    | đại diện cho các giá trị số, ví dụ: 100          |
| <b>Boolean</b>   | đại diện cho giá trị boolean là sai hoặc đúng    |
| <b>Undefined</b> | đại diện cho giá trị không xác định              |
| <b>Null</b>      | đại diện cho null tức là không có giá trị nào cả |

## ❖ Kiểu dữ liệu non-primitive trong javascript

Có năm loại kiểu dữ liệu non-primitive trong **JavaScript**. Gồm:

| Kiểu dữ liệu  | Mô tả   |
|---------------|---|
| <b>Object</b> | chứa nhiều thông tin, kiểu dữ liệu khác nhau, ví dụ<br>{ name: "Nguyen Van A", age: 21} |
| <b>Array</b>  | tập hợp danh sách nhiều giá trị, ví dụ [12, "Nguyen Van A"]                             |
| <b>RegExp</b> | đại diện cho biểu thức chính quy, ví dụ /\w+/   |

## ❖ Giới thiệu về hàm trong Javascript

**Hàm** là một trong những khối xây dựng cơ bản trong **JavaScript**. Một hàm trong **JavaScript** tương tự như một thủ tục – một tập hợp các câu lệnh thực hiện một tác vụ hoặc tính toán một giá trị, nhưng để một thủ tục đủ điều kiện là một hàm, nó sẽ lấy một số đầu vào và trả về đầu ra khi có một số mối quan hệ rõ ràng giữa đầu vào và đầu ra. Để sử dụng một hàm, bạn phải định nghĩa nó ở đâu đó trong phạm vi mà bạn muốn gọi nó.

Có 2 cách tạo hàm phổ biến:

1. Khai báo bằng từ khoá **function**
2. Khai báo bằng **arrow function**

## ❖ Khai báo function

```
function myFunc(theObject) { // Khai báo hàm  
  theObject.make = "Toyota";  
}
```

```
const mycar = {  
  make: "Honda",  
  model: "Accord",  
  year: 1998,  
};
```

```
console.log(mycar.make); // "Honda"  
myFunc(mycar);  
console.log(mycar.make); // "Toyota"
```

Ở đoạn code bên trên, chúng ta khai báo một hàm tên là **myFunc** có chức năng thay đổi giá trị **make** trong obj **myCard** thành "Toyota".

Chúng ta cũng có thể viết hàm **myFunc** theo kiểu **arrow function**

### ❖ Khai báo arrow function

```
const myFunc = (theObject) => { // Khai báo hàm  
  theObject.make = "Toyota";  
}  
....
```

Cách gọi hàm **arrow function** cũng tương tự như cách gọi hàm ở ví dụ bên trên

## ❖ Khởi tạo tham số truyền vào trong function

Ở một số trường hợp, bạn sẽ không muốn tham số truyền vào function của mình là **null** hay **undefine** thì bạn có thể gán giá trị ban đầu cho tham số truyền vào của hàm. Điều này sẽ đảm bảo hàm của bạn luôn hoạt động đúng, theo cách mà bạn xử lý các trường hợp hàm không có giá trị truyền vào khi được gọi

```
const myFunc = (name = "new user") => { // Khai báo hàm  
  console.log("Hello ", name )  
}  
  
console.log(myFunc()); // "Hello new user "  
console.log(myFunc("Nguyen Van A")); // "Hello Nguyen Van A "
```



## ❖ Tham số **rest** truyền vào trong hàm

Tham số **rest** (...) cho phép một hàm xử lý một số lượng đối số không xác định như một mảng:

```
const sum = ({...args}) => {  
  const {a, b, c} = args || {}  
  console.log("Gia tri cua a la ", a)  
  console.log("Gia tri cua b la ", b)  
  console.log("Gia tri cua c la ", c)  
}  
  
sum({a: 10, b: 20, c: 30});
```

Console ×

Gia tri cua a la 10

Gia tri cua b la 20

Gia tri cua c la 30

Ở hàm trên, chúng ta thực hiện truyền vào một object gồm a, b, c. **...args** là biến chứa tất cả các giá trị này

Nếu bạn đã khai báo một biến trong hàm này rồi, thì biến này sẽ không còn trong **rest** nữa

```
const sum = ({b, ...args}) => {  
  console.log("Gia tri cua args la ", args)  
  console.log("Gia tri cua b la ", b)  
}  
  
sum({a: 10, b: 20, c: 30});
```

Console ×

Gia tri cua args la ▶ (2) {a: 10, c: 30}

Gia tri cua b la 20

Bạn có thể thấy trong **args** đã không có giá trị **b** nữa, tại vì đã đã được khai báo ra trước đó, đây là những lưu ý bạn nên cần nắm bắt kỹ.

## ❖ Biểu thức và toán tử

Ở phần này, các bạn sẽ học thêm các biểu thức và toán tử, sau khi học xong bạn sẽ xử lý logic **Javascript** ngắn gọn, dễ hiểu và rút ngắn được thời gian phát triển với ngôn ngữ **Javascript**

Sẽ có rất nhiều biểu thức và toán tử khác nhau, chúng ta không thể nào học hết trong bài học này, thế nên nếu muốn tìm hiểu thêm bạn có thể truy cập trang web <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators>

Sau đây, sẽ là những biểu thức và toán tử thường gặp:

## ❖ Toán tử logical AND (&&)

Toán tử logic **AND (&&)** (kết hợp logic) gồm các điều kiện boolean, sẽ return **true** nếu và chỉ khi tất cả các điều kiện là **true**. Nếu không nó sẽ **false**.

```
const a = 3;  
const b = -2;  
  
console.log(a > 0 && b > 0);  
// Kết quả: false
```

## ❖ Toán tử logical OR (||)

Toán tử logic **OR (||)** gồm các điều kiện return **true** nếu và chỉ khi một hoặc nhiều điều kiện của nó là **true**. Nó thường được sử dụng với các giá trị boolean (logic). Khi đó, nó trả về một giá trị Boolean. Tuy nhiên, || toán tử thực sự trả về giá trị của một trong các toán hạng được chỉ định, vì vậy nếu toán tử này được sử dụng với các giá trị không phải Boolean, nó sẽ trả về giá trị không phải Boolean.

```
const a = 3;  
const b = -2;  
  
console.log(a > 0 || b > 0); // Kết quả: true
```

## ❖ Toán tử logical NOT (!)

Toán tử **NOT (!)** là giá trị phủ định của biến. Nó thường được sử dụng với các giá trị boolean (logic). Khi được sử dụng với các giá trị không phải Boolean, nó trả về false nếu toán hạng đơn của nó có thể được chuyển đổi thành true; nếu không, trả về true.

```
const a = 3;  
const b = -2;  
console.log(!(a > 0 || b > 0)); // Kết quả: false
```

## ❖ Optional chaining (?.)

Toán tử **optional chaining (?.)** truy cập thuộc tính của một đối tượng hoặc gọi một hàm. Nếu đối tượng được truy cập hoặc hàm được gọi bằng toán tử này là **undefined** hoặc **null**, biểu thức ngắn mạch và đánh giá là không xác định thay vì gây lỗi cho chương trình

```
const adventurer = {  
  name: 'Alice',  
  cat: { name: 'Dinah' },  
};  
  
console.log(adventurer.dog?.name); // Kết quả: undefined  
console.log(adventurer.dog?.name);  
// Kết quả: Error: Cannot read properties of undefined (reading 'name')
```

Bạn có thể thấy ở **console.log** thứ nhất giá trị được xuất ra là **undefined** bởi vì không có giá trị **dog** nào ở đây cả

Ở **console.log** thứ hai, đã xảy ra lỗi, đây là thứ chúng ta không mong muốn. Đặc biệt đối với lập trình ứng dụng di động đa nền tảng (React Native), điều này sẽ gây crash app (ứng dụng dừng hoạt động và văng ra ngoài màn hình chính).

Một số **optional chaining** khác

```
arr?.[1, 2, 3]  
func?.()
```



- ☐ **Hiểu về React Native**
- ☐ **Phạm vi ứng dụng của React Native**
- ☐ **Xu hướng phát triển của React Native**
- ☐ **Thiết lập được môi trường lập trình React Native trên Window và MacOS**
- ☐ **Ôn lại kiến thức javascript cơ bản**



**Kết thúc**