TRƯỜNG ĐẠI HỌC CÔNG NGHỆ SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**

**THỰC TẬP TỐT NGHIỆP**

*Chủ đề:*

**Tìm hiểu công nghệ Nodejs.**

**Sinh viên thực hiện:**

1. Võ Hoàng Kỳ MSSV: DH51704991 Lớp:D17-TH04

# LAB 1: NGÔN NGỮ JAVASCRIPT VÀ NODEJS

## Mục tiêu:

* Hiểu và nắm vững kiến thức về ngôn ngữ javascript.
* Hiểu được cách thức hoạt động Nodejs.

## Mục tiêu:

* Biết cách xây dựng hàm.
* Hiểu và sử dụng được các cấu trúc rẽ nhánh.

## Tóm tắc lý thuyết:

* 1. Ngôn ngữ javascript: JavaScript là một ngôn ngữ lập trình của HTML và WEB. Nó sử dụng phổ biến nhất như là một phần của các trang web, mà sự thi hành của chúng cho phép Client-Side script tương tác với người sử dụng và tạo các trang web động. Nó là một ngôn ngữ chương trình thông dịch với các khả năng hướng đối tượng. Javascript là một ngôn ngử hoạt dộng bất đồng bộ, có thể thực hiện nhiều câu l;ệnh cùng 1 thời gian.
  2. Function trong javascript:
     + Kiểu dữ liệu: có 6 kiểu dữ liệu cơ bản ( Number, Boolean, String, Null, Underfined, Object)
     + Khai báo hàm: có 2 cách khai báo hàm trong javascript

**function** *<Tên hàm>* **(***<Danh sách các tham số>***){**

*//…code các câu lệnh*

**}**

hoặc

**const** *<Tên bến>* **= function(***<Danh sách các tham số>***){**

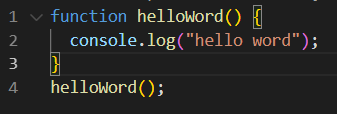
*//…code các câu lệnh*

**}**

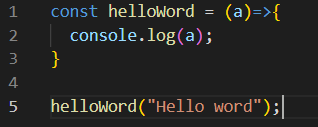
* 1. Nodejs là một nền tảng được xây dựng trên V8 JavaScript Engine – trình thông dịch thực thi mã JavaScript (chrome V8), giúp xây dựng các ứng dụng web một cách đơn giản và dễ dàng mở rộng. Nodejs là một runtime (run-time system : phần mềm được thiết kế để chạy các chương trình máy tính với một hoặc nhiều ngôn ngữ xác định).
  2. Cài đặt Nodejs: Download tại https://nodejs.org/en/download/ chọn đúng hệ điều hành và phiên bản để tải xuống.

## Vận dụng:

1. Sử dụng hàm và biến trong javascript. (thực hiện debug trên chrom console)
   1. Biểu thức hàm:

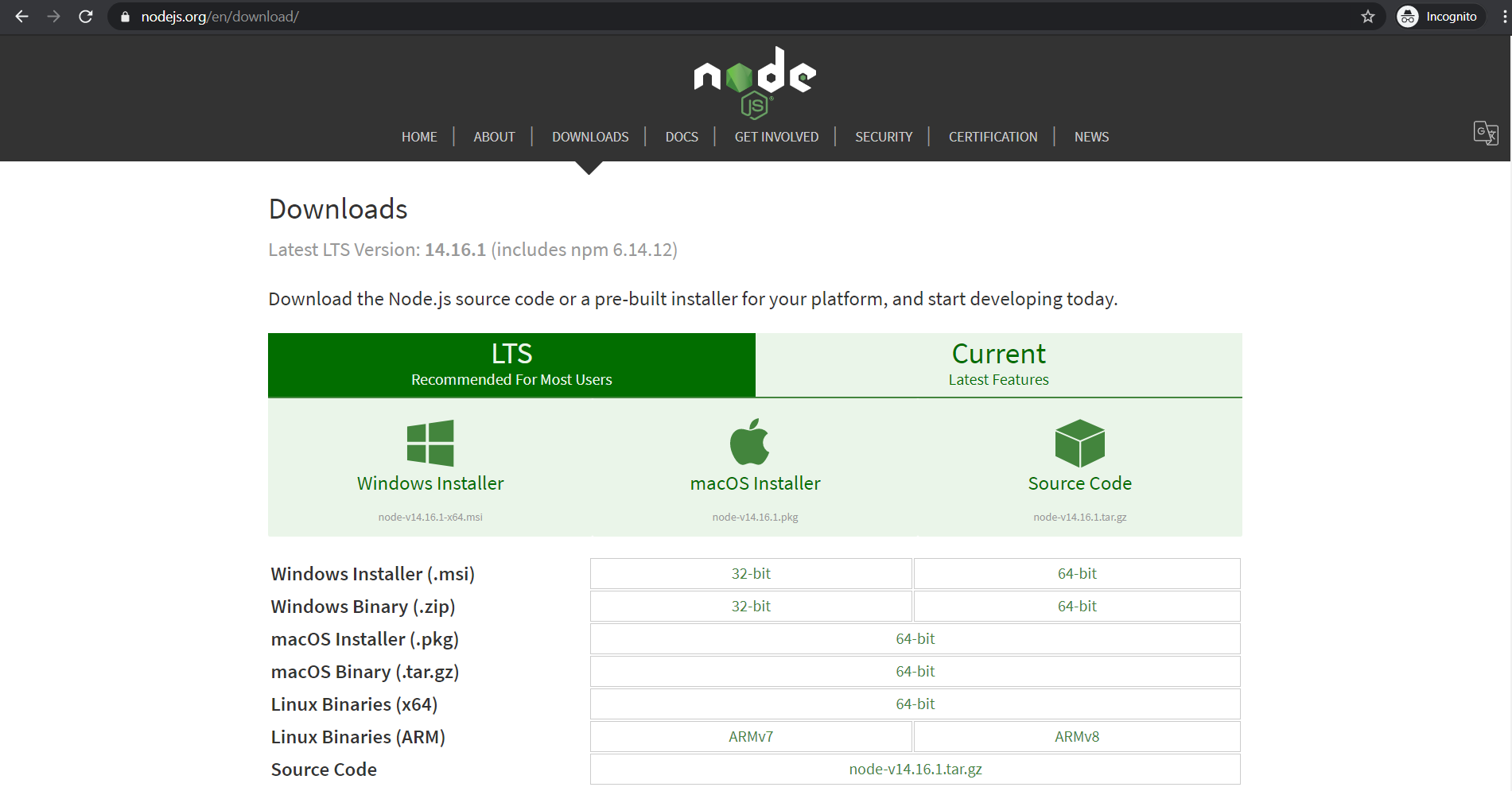


* + - Dòng 1 đến dòng 3 dùng xây dựng hàm.
    - Dòng 4 dùng thực thi các lệnh trong hàm, và kết quả: 
  1. Hàm trong biến:

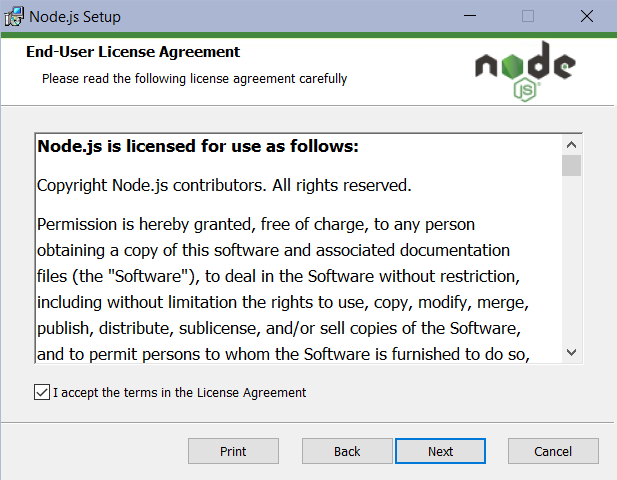


* + - Dòng 1 đến dòng 3 dùng xây dựng hàm.
    - Dòng 5 dùng thực thi các lệnh trong hàm. Và kết quả: 

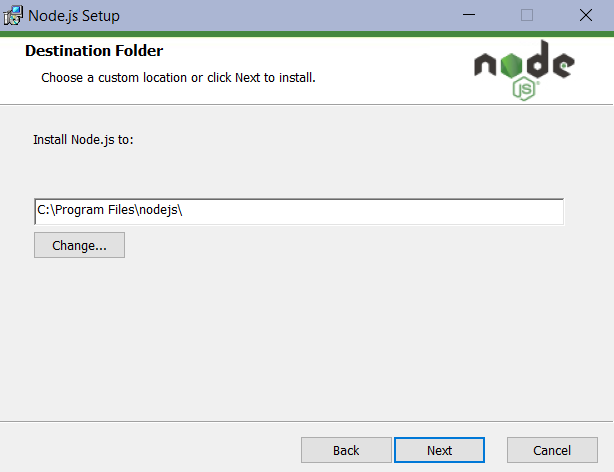
1. Cài đặt Nodejs trên window 10:
   * + Bước 1: Truy cập trang https://nodejs.org/en/download/.



* + - Bước 2: chọn windows installer và tải về mở file cài đặt nhấn **next**
    - Bước 3: nhấn tích vào ô chọn **next**.



* + - Chọn ổ đĩa lưu trữ (khuyên dùng ổ mặc định).



* + - Bước 5: Chọn **next-> install-> finish**. Hoàn tất cài đặt môi trường nodejs trên window 10.
    - Bước 6: kiểm tra phiên bản nodejs với lệnh “ *node -v* “. Phiên bản mới nhất hiện tại là 14.16.1.



# LAB 2: CÀI ĐẶT NODEJS VÀ CHƯƠNG TRÌNH ĐẦU TIÊN

## Mục tiêu:

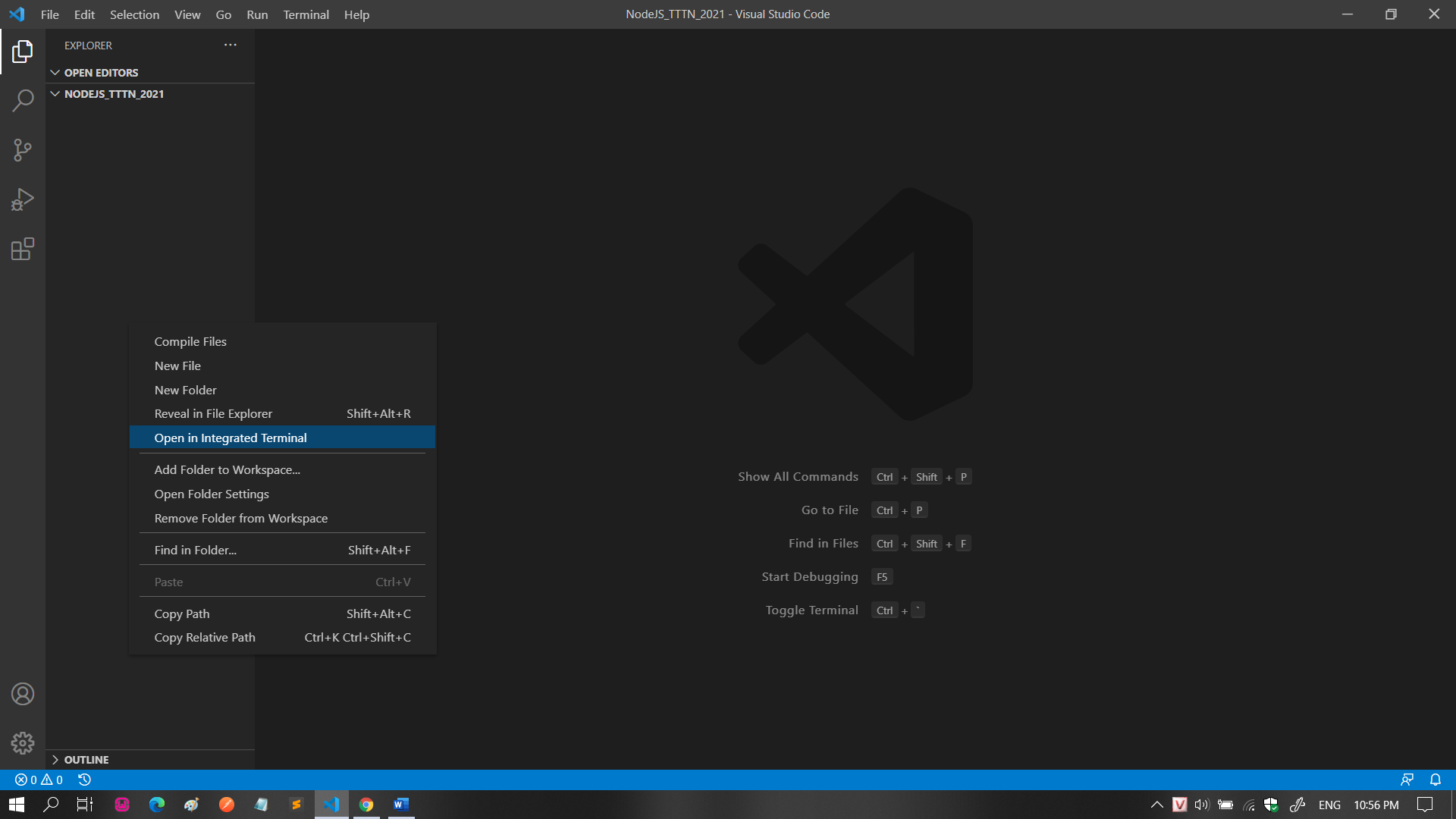
* Biết NPM (node packed manager).
* Biết cài đặt và tạo chương trình Nodejs.
* Thực hiện cài đặt các thư viện frameword và chạy chương trình Nodejs.

## Tóm tắc lý thuyết:

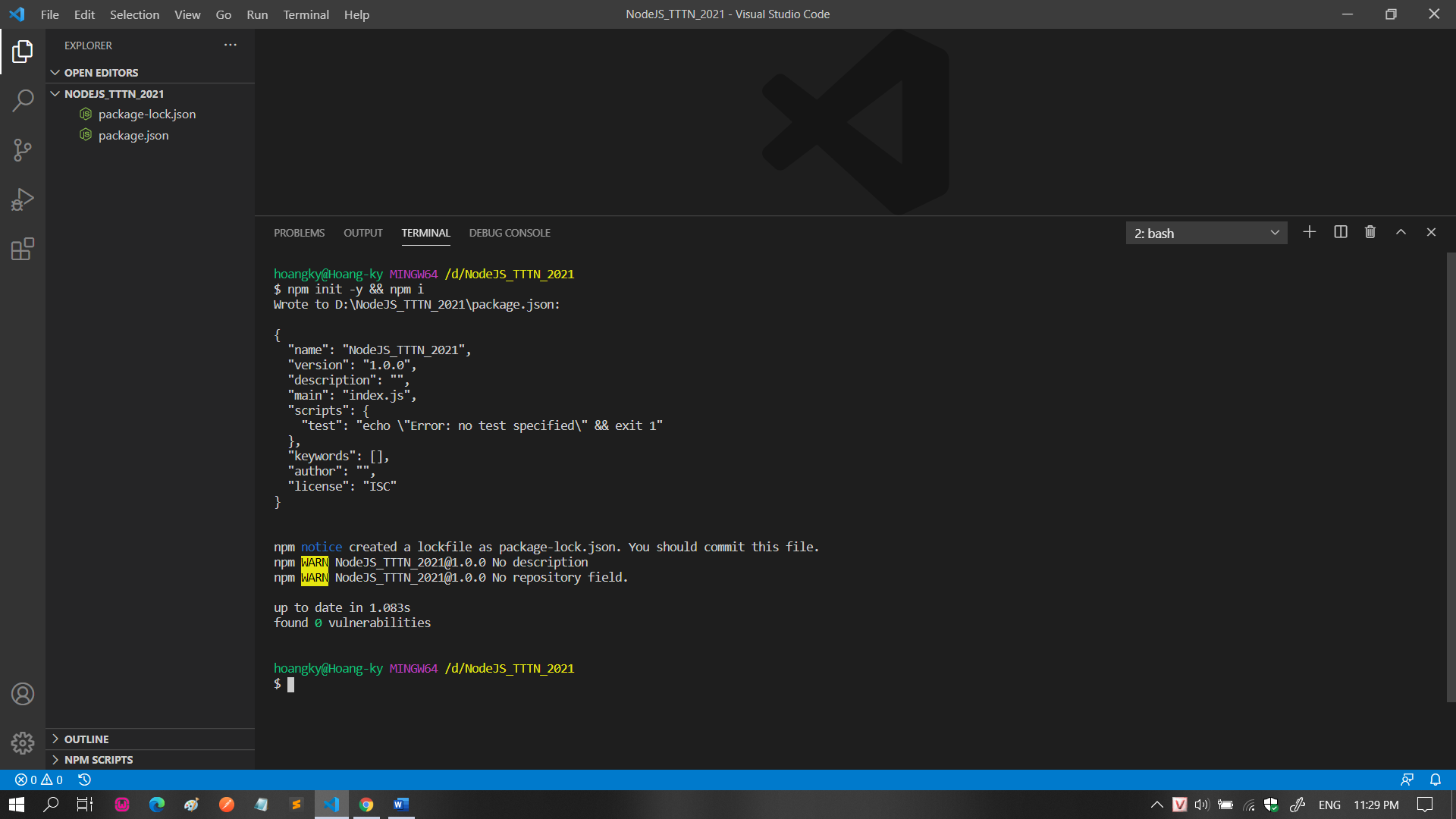
1. NPM (node packed manager) là một công cụ tạo và quản lý các thư viện lập trình Javascript cho Node.js. Trong cộng đồng Javascript, các lập trình viên chia sẻ hàng trăm nghìn các thư viện với các đoạn code đã thực hiện sẵn một chức năng nào đó. Nó giúp cho các dự án mới tránh phải viết lại các thành phần cơ bản, các thư viện lập trình hay thậm chí cả các framework.
2. Cấu trúc một project nodejs bắt buộc cần phải có folder node\_modules, file package-lock.json, file package.json và 1 file javascript.

## Vận dụng:

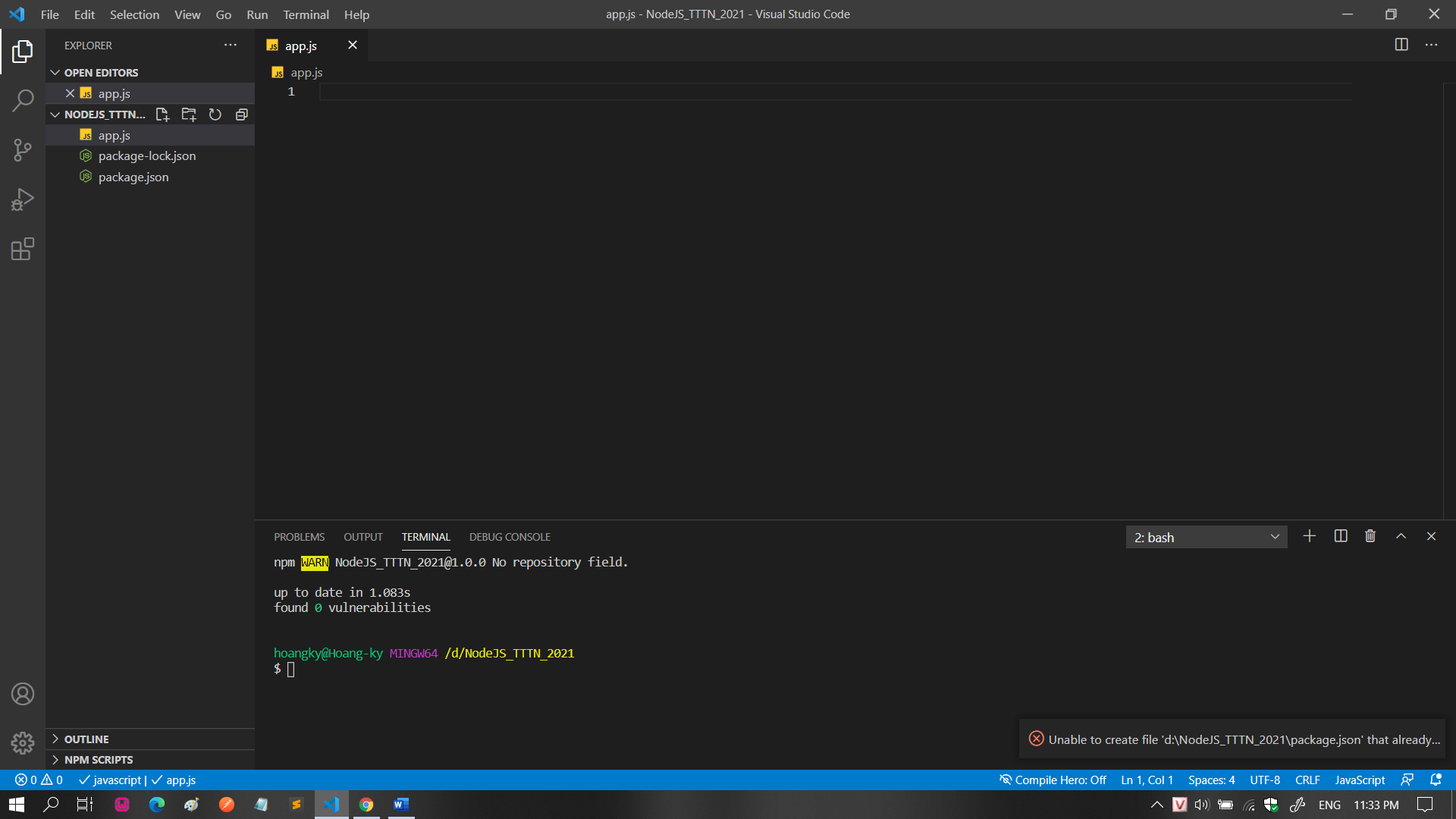
* + - 1. Cài đặt NPM và tạo project dùng Nodejs (thực hiện trên editor Visual Code)
    - Tạo thư mục mở thư mục trong visual code.
    - Nhấn chuột phải vào folder chọn **Open integrated terminal.**



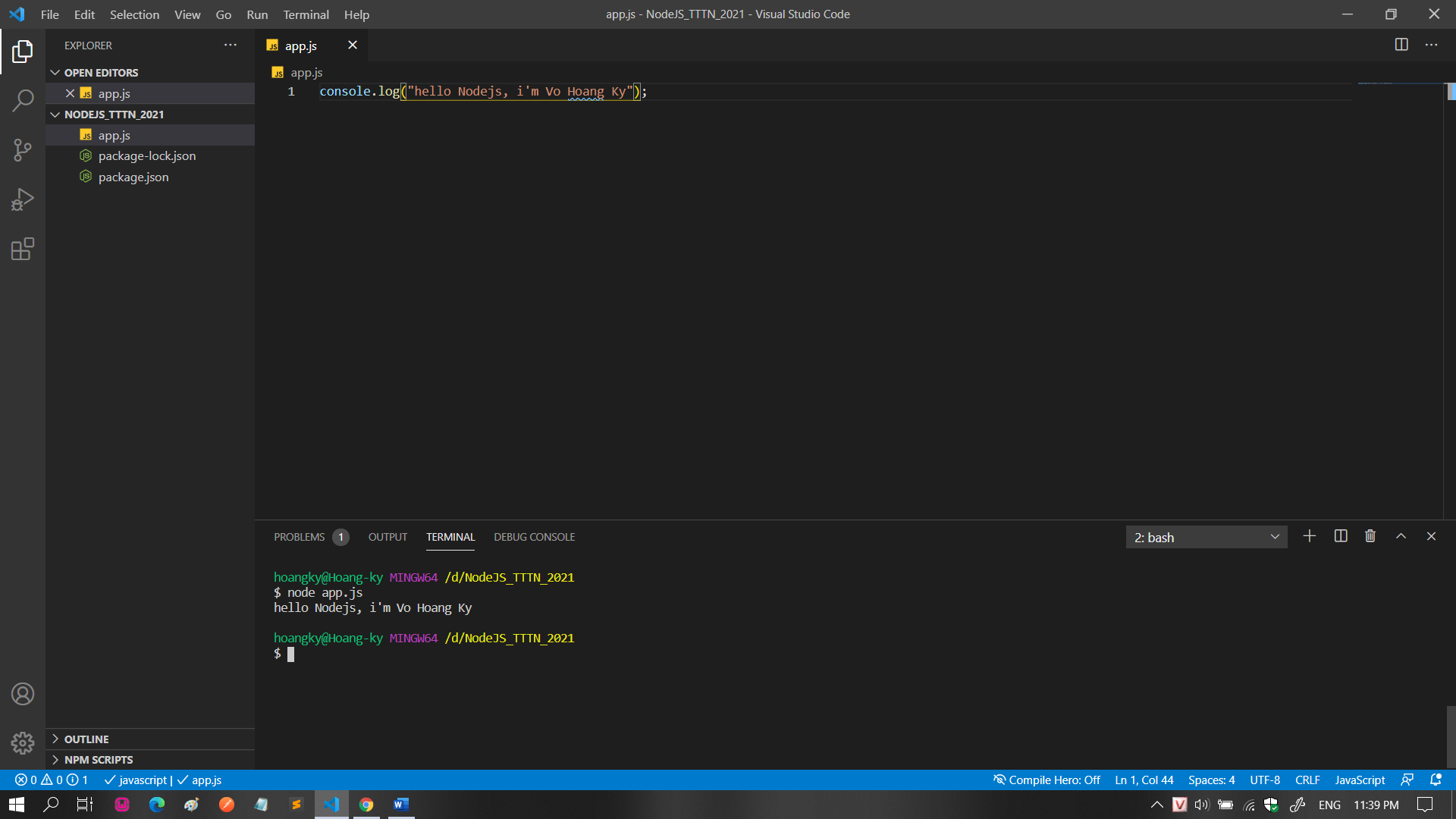
* + - Chọn vào vùng terminal gõ “ *npm init* -y *&& npm i*“



* + - Tạo một file javascript ( app.js)



* + - Thực hiện test thử chương trình trong file app.js bằng lệnh   
      *“node app.js”*



# LAB 3: BIẾN HẰNG VÀ CÁC TOÁN TỬ TRONG NODEJS

## Mục tiêu:

* Nắm được khái niệm về biến và hàm trong Nodejs.
* Biết được cách khai báo và nguyên tắc đặt tên biến, hằng trong Nodejs.
* Hiểu và biết cách sử dụng một số toán tử trong Nodejs.

## Tóm tắt lí thuyết:

1. Biến dữ liệu trong Nodejs được thực hiện khai báo bằng các từ khóa **var**, **let**, **const**. Có 4 cách khai báo biến trong Nodejs:

**var** *<tên biến>* **=** *<giá trị>*

hoặc

**let** *<tên biến>* **=** *<giá trị>*

hoặc

**const** *<tên biến>* **=** *<giá trị>*

1. Phân biệt giữa var, let và const

* **var** là kiểu khai báo biến toàn cục và có thể thay đổi giá trị, kiểu dữ liệu tại mọi vị trí trong chương trình.
* **let** là kiểu khai báo biến cục bộ của một hàm, 1 đoạn chương trình. Giá trị khai báo bằng từ khóa **let** có thể cập nhật nhưng không thể tái khởi tạo trong hàm, đoạn chương trình.
* **const** là kiểu khai báo biến toàn cục nhưng giá trị của chúng được mặc định và không thể thay đổi sau khi khai báo.

1. Hằng số trong Nodejs tương tự như javascript, hằng số chỉ chứa được một giá trị nhất định nhận được sau khi khởi tạo và không thay đổi.
2. Nguyên tắc đặt tên hằng và biến trong nodejs:

* Tên hằng và biến chỉ được bắt đầu bằng kí tự “ \_ “ hoặc chữ.
* Không được đặt tên hằng, biến có kí tự đầu là số.
* Trong tên hằng và biến không chứa các kí tự: !@#$%^&\*()-

1. Các toán tử trong nodejs :

* Toán tử gán: Toán tử thường được sử dụng để khởi tạo giá trị hoặc thay đổi giá trị của một biến, cú pháp

*<tên biến>* **=** *<giá trị>*

* Toán tử số học là một dạng các phép tính giản đơn cộng, trừ, nhân, chia, phép chia lấy dư trong số học.

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Mô tả** | **Cú pháp** |
| + | Cộng 2 số hạng | *<giá trị 1>* **+** *<giá trị 2>* |
| - | Trừ 2 số hạng | *<giá trị 1>* **-** *<giá trị 2>* |
| \* | Nhân 2 số hạng | *<giá trị 1>* **\*** *<giá trị 2>* |
| / | Chia 2 số hạng | *<giá trị 1>* **/** *<giá trị 2>* |
| % | Phép chia lấy dư 2 số hạng | *<giá trị 1>* **%** *<giá trị 2>* |

* Toán tử so sánh là toán tử dùng để thực hiện các phép toán so sánh giữa 2 số hạng có giá trị và có kết quả trả về là **true** hoặc **false**.

|  |  |
| --- | --- |
| **Toán tử** | **Mô tả** |
| == | Hai số hạng có giá trị bằng nhau |
| === | Hai số hạng bằng nhau và cùng chung kiểu dữ liệu |
| != | Hai số hạng khác nhau về giá trị |
| !== | Hai số hạng khác nhau về giá trị hoặc kiểu dữ liệu |
| > | Số hạng bên trái lớn hơn số hạn bên phải |
| >= | Số hạng bên trái lớn hơn hoặc bằng số hạn bên phải |
| < | Số hạng bên trái nhỏ hơn số hạn bên phải |
| <= | Số hạng bên trái lớn nhỏ hoặc bằng số hạn bên phải |

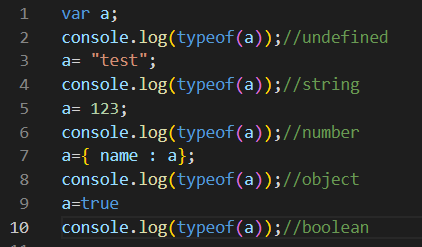
* Toán tử logic là tổ hợp các giá trị boolean

|  |  |  |  |
| --- | --- | --- | --- |
| **Toán tử** | **Tên** | **Ví dụ** | **Kết quả** |
| || | Hoặc ( or ) | True || false | true |
| && | Và ( and ) | True && false | False |
| ! | Nghịch đảo ( not ) | !True | false |

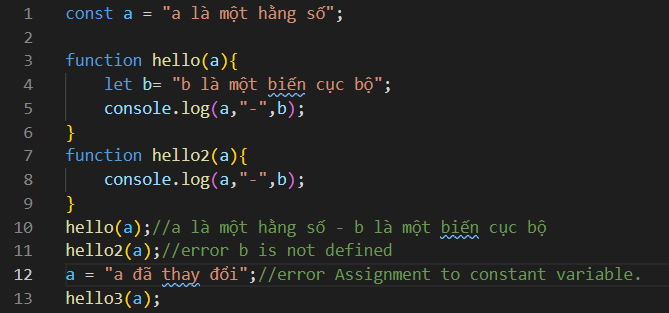
## Vận dụng

1. Khởi tạo một biến xem kiểu dữ liệu và phạm vi sử dụng của chúng.
   * + Xem và xác định kiểu dữ liệu của một biến ta sử dụng từ khóa

“ **typeof(*<tên biến>*)** ”



* + - Xác định phạm vi sử dụng biến



Dòng 1 khai báo 1 hằng số a.

Dòng 3-6 tạo một hàm **hello** có sử dụng biến biến **b** khởi tạo kiểu let

Dòng 7-9 tạo 1 hàm **hello2** không khai báo biến cục bộ **b**

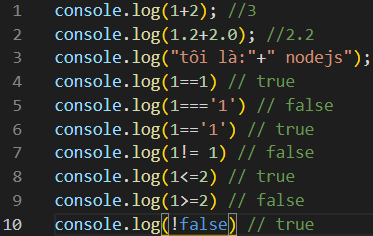
Dòng 10 thực hiện hàm **hello** có truyền hằng số **a** và nhận kết quả là:

“ *a là một hằng số - b là một biến cục bộ* ”

Dòng 11 thực hiện hàm **hello2** sẽ bị lỗi “ *b is not defined ”* ( biến b không xác định)

Dòng 12 thay đổi giá trị hằng số a sẽ nhận thông báo lỗi và không thể thwujc hiện dòng 13.

1. Sử dụng một số toán tử trong nodejs và nhận xét kết quả:



Dòng 1 phép cộng 2 số nguyên

Dòng 2 phép cộng 2 số thực

Dòng 3 phép cộng chuỗi

Dòng 4 phép so sánh 2 số nguyên ( cùng kiểu dữ liệu)

Dòng 5 phép so sánh giá trị và kiểu dữ liệu 2 số nguyên (khác kiểu dữ liệu)

Dòng 6 phép so sánh giá trị 2 số nguyên (khác kiểu dữ liệu)

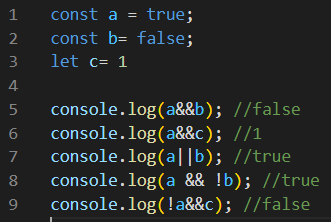
Dòng 7 kiểu tra sự khác nhau giữ 2 giá trị

Dòng 8 so sánh bé hơn hoặc bằng

Dòng 9 so sánh lớn hơn hoặc bằng

Dòng 10 nghịch đảo giá trị boolean

1. Một số toán tử logic và biến



Dòng 1 tạo hằng số a giá trị true

Dòng 2 tạo hằng số b giá trị là false

Dòng 3 tạo biến c giá trị 1

Dòng 5 logic a và b kết quả là false

Dòng 6 logic a và c kết quả là 1

Dòng 7 logic a hoặc b kết quả là true

Dòng 8 logic a và nghịch đảo b kết quả là true

Dòng 9logic nghịch đảo a và c kết quả là false.

# LAB 4: HÀM VÀ CÁC CÂU LỆNH RẼ NHÁNH TRONG NODEJS

## Tóm tắt lí thuyết:

* Hàm trong nodejs được xây dựng và sử dụng tương tự như javascript.
* Các cấu trúc rẽ nhánh:
  + - If-else là một loại điều kiện được sử dụng rất nhiều trong các ngôn ngữ lập trình. if-else sẽ hoạt động như

**if( *<điều kiện>* ){**

***//code. Khi điều kiện là đúng sẽ thực hiện***

**}**

**else{**

***//code. Khi điều kiện là sai sẽ thực hiện***

**}**

*Hoặc*

**(*<điều*** ***kiện>*)? ( //*code khi điều kiện đúng*) : (//*code khi điều kiện sai*)**

* + - Switch – case

**Switch (*<biến giá trị>*)**

**{**

**Case *<giá trị 1>*:**

**//code hanh dong;**

**Break;**

**…………**

**Case *<giá trị N>*:**

**//code hanh dong;**

**Break;**

**Default:**

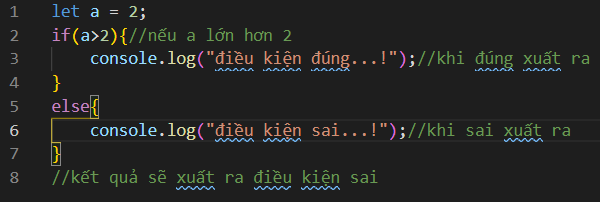
**// code hanh dong;**

**Break;**

**}**

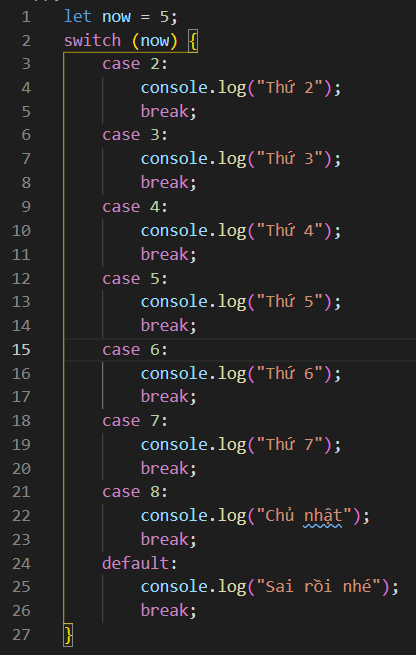
## Vận dụng

1. If-else



Kết quả đoạn code trả về là: “ điều kiện sai…!”

1. Switch-case



# LAB 5: ĐỐI TƯỢNG-OBJECT TRONG NODEJS

## Mục tiêu:

* Biết được Object trong nodejs.
* Cách khai báo và tạo một đối tượng trong nodejs.

## Tóm tắt lí thuyết:

* Trong nodejs đối tượng là một khái niệm trừu tượng thể hiện cho một đối tượng cụ thể mà chúng ta có thể tự tạo một đối tượng theo ý của mình dựa vào yêu cầu của ứng dụng.
* Để khởi tạo một đối tượng trong nodejs có 2 cách sau:
  + - Sử dụng từ khóa new Object()

***<kiểu biến> <tên đối tượng>* = new Object();**

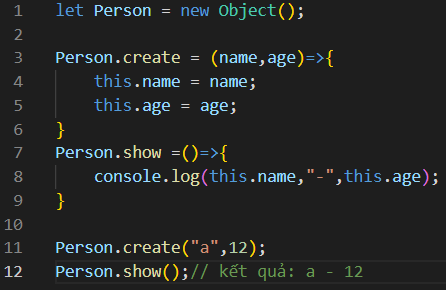
* + - Sử dụng từ khóa {}

***<kiểu biến> <tên đối tượng>* = { };**

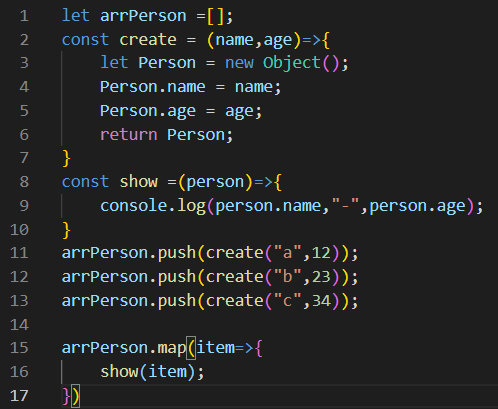
* Thuộc tính của một đối tượng là những đặt điểm cần lưu trữ.
* Phương thức trong hướng đối của nodejs cũng có các tính chất như các ngôn ngữ khác, chúng ta có thể hiểu là hàm.

## Vận dụng:

* + - 1. Khởi tạo một đối tượng **Person**, xây dựng các thuộc tính và phương thức.



* + - 1. Từ Object Person xây dựng 1 mảng chứa nhiều đối tượng và sử dụng show để hiển thị



Kết quả khi thực hiện show:



# LAB 6: KHỞI TẠO SERVER NODEJS VỚI HTTP MODULE

## Mục tiêu:

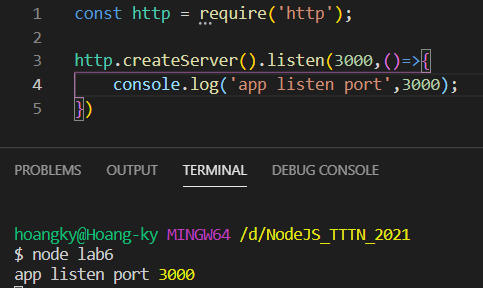
* Biết về HTTP module.
* Khởi tạo server với HTTP module.

## Tóm tắt lí thuyết:

* Trong HTTP là một module được tích hợp sẵn vào trong Node.js (nên sẽ không cần phải download), module này có nhiệm vụ khởi tạo một cổng kết nối HTTP server trả về client.
* Để sử dụng được bất kỳ module nào trong Node.js thì chúng ta cần phải require module đó. Và để require một module trong Node.js chúng ta sử dụng từ khóa require.

## Vận dụng:

1. Khởi tạo một server .

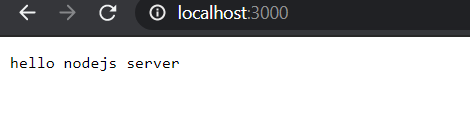


Đoạn code thực hiện việc tạo một sever sử dụng port 3000.

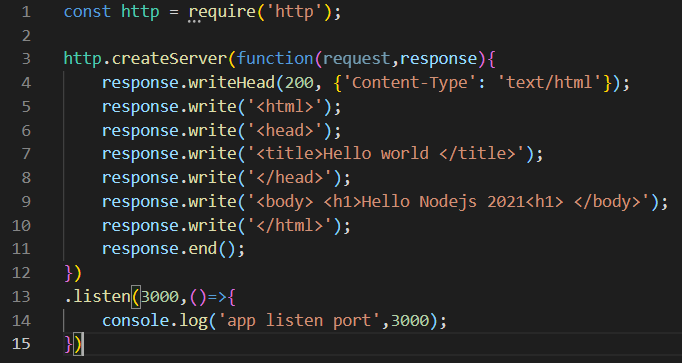
1. Làm quen với Request và Response trong sever-client



* + - Khi khởi chạy với đoạn chương trình ta sẽ được kết quả trên trình duyệt:



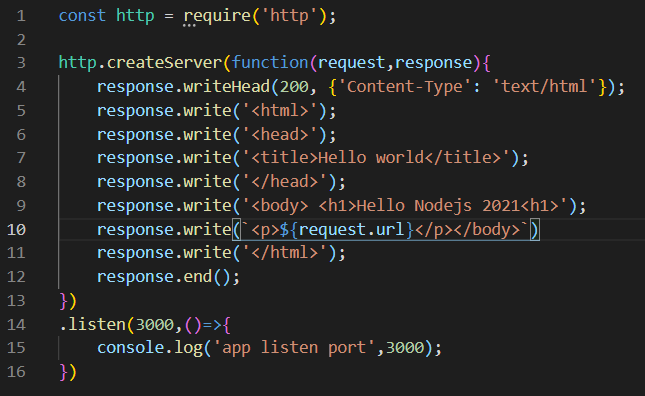
1. Thiết lập head từ sever



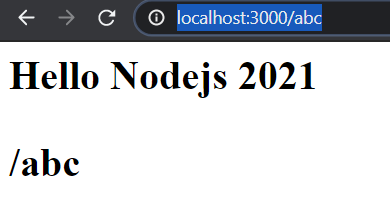
* + - Sau khi chạy đoạn chương trình ta được kết quả:



1. Thuộc tính Request và paramter trong URL



* + - Thực hiện chạy đoạn chương trình với đường dẫn <http://localhost:3000/abc>



* + - Đường dẫn “/abc” được xuất ra từ dòng 10 của đoạn chương trình.

# LAB 7: ĐỌC GHI FILE VỚI MODULE FS TRONG NODE.JS

## Mục tiêu:

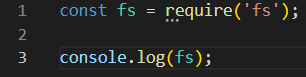
* Biết được sử dụng module fs.
* Thực hiện được đọc ghi file trong Nodejs.

## Tóm tắt lí thuyết:

* Module fs (viết tắt của file system) là một module được tích hợp sẵn trong node.js có chức năng xử lý file, thư mục trong nodejs.
* Module fs này chứa rất nhiều các function dùng để xử lý file và thư mục.

## Vận dụng:

* + - 1. Khai báo module fs.



Dòng 3 dùng để xem 1 số hàm chức năng có trong module fs

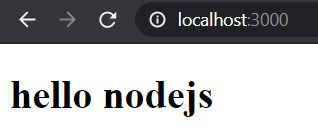
* + - 1. Đọc file html với module fs và hiển thị trên trình duyệt.
    - Tạo 1 file lab7.html



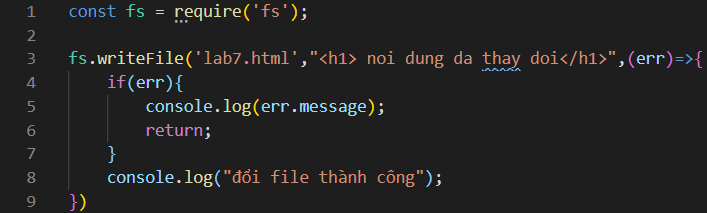
* + - Thực hiện xây dựng chương trình



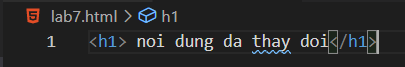
* + - Chạy đoạn chương trình và nhạn kết quả



1. Ghi file với module fs.



* + - Thực hiện đoạn chương trình sẽ nhận được kết quả tail file lab7.html



# LAB 8: PHÂN TÍCH URL VỚI MODULE URL TRONG NODEJS

## Mục tiêu:

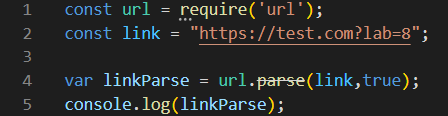
* Hiểu về module url trong nodejs.
* Thực hiện khai báo và sử dụng module url trong nodejs.

## Tóm tắt lí thuyết:

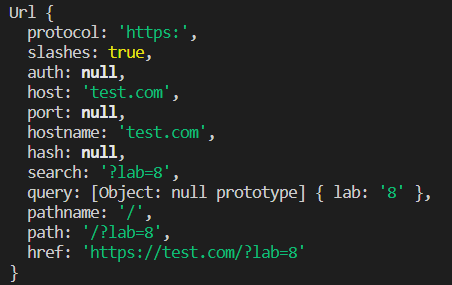
* Module url à một module được tích hợp sẵn vào trong core của node.js, nó được dùng để xử lý và phân tích chuỗi URL dựa vào đó chúng ta có thể biết được các thông số của URL đó.
* Module có các thuộc tính giống hệt như đối tượng location trong javascript.

## Vận dụng:

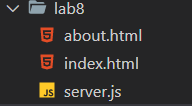
1. Khai báo và sử dụng một số chức năng module url trong nodejs.



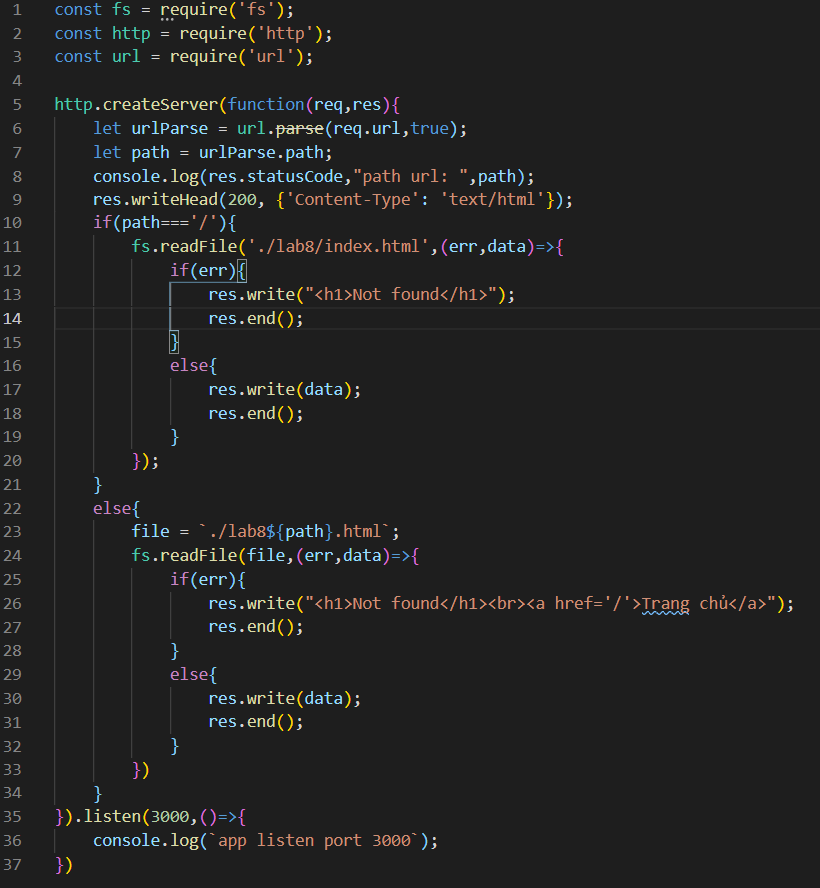
Thực hiện chạy đoạn chương trình ta được kết quả là 1 đối tượng url với các thuộc tính:



1. Kết hợp ba module HTTP, fs và URL xây dựng một websever.
   * + Xây dựng thư mục gồm các file



* + - Xây dựng server



* + - Vào thư mục lab8 và khởi chạy chương trình với lệnh: *node server.js*





# LAB 9: NPM TRONG NODE.JS

## Mục tiêu:

* Hiểu về npm trong nodejs.
* Biết cách cài đặt package với npm.
* Xây dựng project nodejs vơi npm.

## Tóm tắt lí thuyết:

* Npm (Node.js Project Manage) là một chương trình quản lý thư viện, source của node.js nó được tích hợp sẵn vào trong node.js. Nên khi các bạn cài đặt node.js thì cũng đồng nghĩa với việc các bạn đã cài npm..
* Npm cũng giống như composer trong PHP, nó sử dụng command line để quản lý (download, update, delete) các package cho node.js.
* Để khởi tạo một dự án node.js với npm chúng ta sử dụng lệnh *npm init*
* Để cài đặt một package mới trong NPM chúng ta sử dụng câu lệnh npm install với cú pháp **npm install *tenPackge@version option .***Trong đó :
  + - ***tenPackge*** là tên của package mà các bạn cần cài.
    - ***version*** là version của package mà các bạn muốn cài (nếu để trống nó sẽ tải bản mới nhất).
    - ***option*** là các thông số thêm:

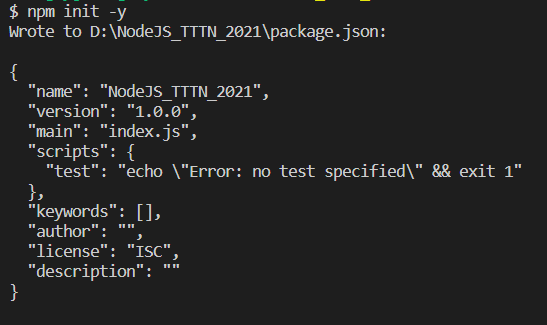
**-S, --save**: Package sẽ được thêm vào ***dependencies*** trong file ***package.json***.

-D, --save-dev: Package sẽ được thêm vào devDependencies trong file ***package.json***.

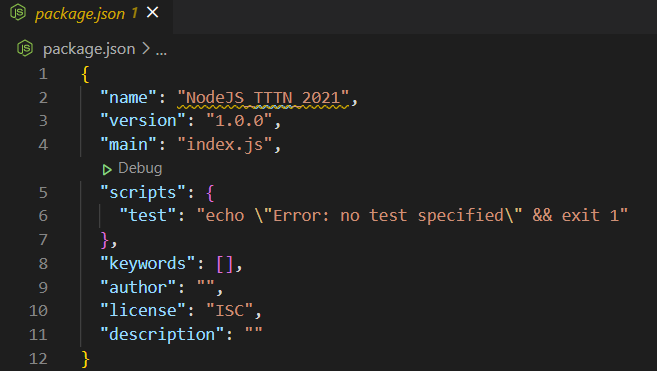
**-O, --save-optional**: Package sẽ được thêm vào ***optionalDependencies*** trong file ***package.json***.

## Vận dụng:

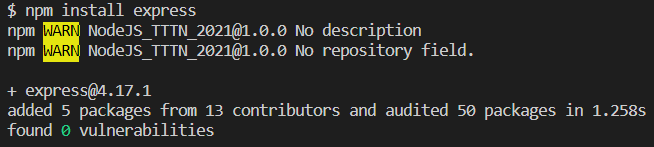
1. Tạo project nodejs với NPM
   * + Thực hiện khởi tạo NPM bằng lệnh *“npm init -y”*



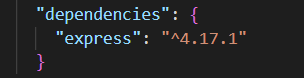
* + - Sau khi khởi tạo thành công ta sẽ nhận được một file package.json



1. Thực hiện cài đặt express framework
   * + Để cài đặt express framework ta thực hiện với lệnh   
       *“npm install express”*.



* + - Sau khi cài đặt thành công trong file package.json sẽ hiện phiên bản và framework đã cài



# LAB 10: UPLOAD FILE TRONG NODE.JS

## Mục tiêu:

* Hiểu và sử dụng module formidable.
* Xây dựng mô hình sever-client upload file.

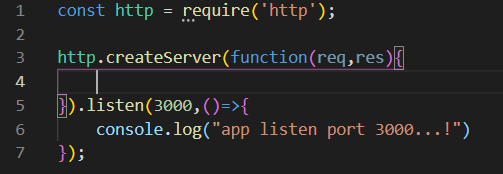
## Tóm tắt lí thuyết:

* Để thực hiện được chức năng này thì mọi người cần phải có các kiến thức sau:
  + - module http trong node.js.
    - module fs trong node.js.
    - module formidable - Đây là một module dùng để phân tích dữ dữ liệu từ form gửi lên, đặc biệt là với tệp tin.
* Module formidable không được tích hợp sẵn vào trong node.js nên để có thể sử dụng được thì các bạn cần phải tải nó về bằng npm. Cú pháp:

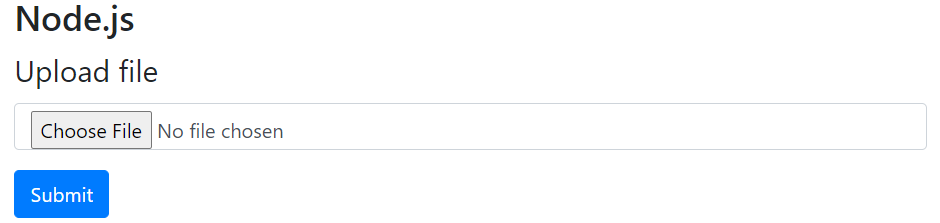
*“npm install -s formidable”*

## Vận dụng:

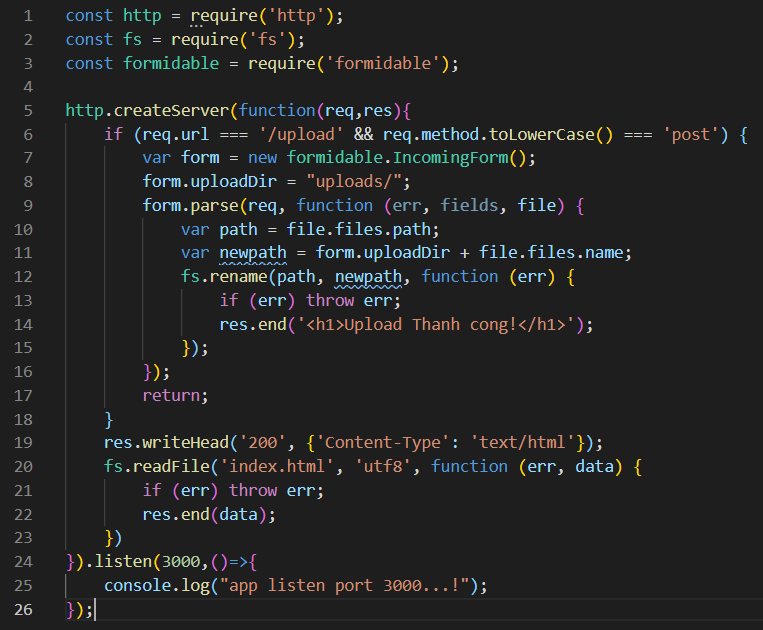
* + - * 1. Xây dụng mô hình server-client side upload file bằng formidable
    - Khởi tạo 1 server bằng module http.



* + - Xây dựng 1 giao diện người dùng



* + - Xây dựng code server cho việc upload file



* + - Thực hiện chạy server bằng lệnh *“node server.js”*. chương trình sẽ thực hiện tại địa chỉ [*http://localhost:3000/*](http://localhost:3000/)

# LAB 11: EVENT TRONG NODE.JS

## Mục tiêu:

* Hiểu về event loop trong nodejs.
* Biết thêm và thực thi sự kiện.

## Tóm tắt lí thuyết:

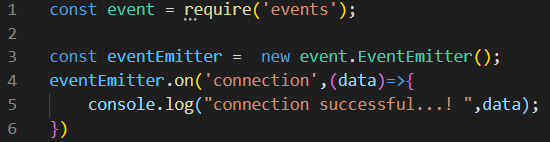
* **Event loop:** Node.js sử dụng rất nhiều sự kiện, nên đó cũng là 1 trong các yếu tố khiến cho node.js thực thi nhanh hơn các ngôn ngữ khác. Khi node.js bắt đầu chạy thì nó hoàn toàn chưa có gì xảy ra, cho đến khi nó bắt được các sự kiện thì nó mới thực thi code cho từng sự kiện và các sự kiện đó có thể sẽ lặp đi lặp lại trong suốt quá trình thực thi ứng dụng.
* Để sử dụng được event trong Node.js thì chúng ta cần phải require module **event** và khởi tạo đối tượng **eventEimitter**. Sau khi thêm ta thực thi sự kiện bằng phương thức **emit**.
* Cú pháp:
* “*on(eventName, handleEvent);*”
* “*emit(evenName[,...args]);*”

Trong đó:

* *eventName* là tênsự kiện muốn thêm.
* *handleEvent* là hàm xử lý khi eventName được gọi. handleEvent có thể là một callbackfunction hoặc là một hàm ẩn danh (closure function).
* *Args*  là các tham số mà bạn muốn truyền vào sự kiện.

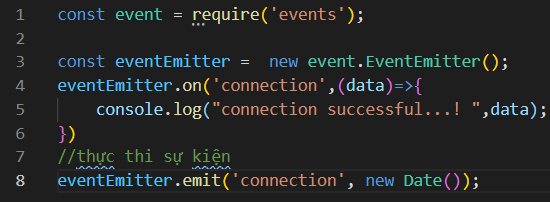
## Vận dụng

1. Khai báo và thêm một sự kiện:



* + - Thêm sự kiện **eventEmitter** với sự kiện **connection.**

1. Thực thi sự kiện:
   * + Sử dụng phương thức emit để thực thi sự kiện



* + - Kết quả nhận được khi chạy chương trình xuất ra chuỗi sự kiện ngày giờ hiện tại



# LAB 12: GỬI MAIL TRONG NODE.JS

## Mục tiêu:

* Biết về module nodemailer.
* Khai báo và cấu hình module nodemailer.

## Tóm tắt lí thuyết:

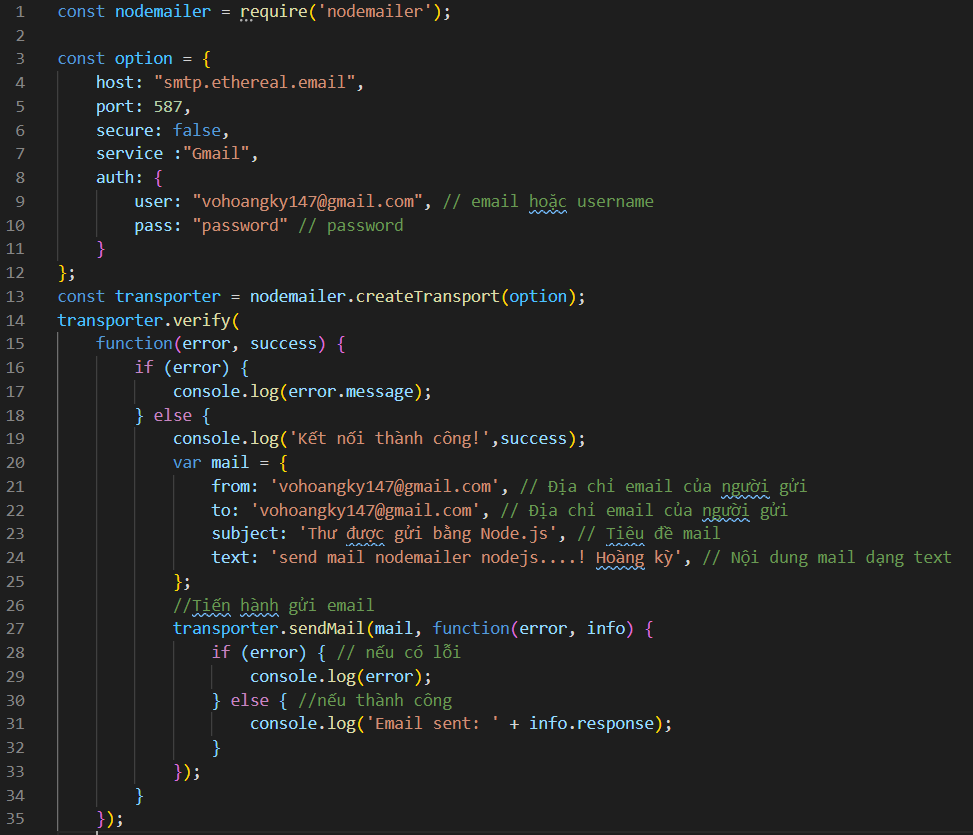
* Nodemailer là một module có trong nodejs dùng để thực hiện việc gửi mail. Nodemailer không có sẵn trong thư viện nodejs cần được cài đặt bằng **npm.**

|  |  |
| --- | --- |
| **Field** | **Mô Tả** |
| **sender** | Email sẽ xuất hiện trong sender |
| **replyTo** | Email sẽ nhận thư trả lời. |
| **Encoding** | Dạng encode text/html. Mặc định là utf-8. |
| **attachments** | Object chứa các file đính kèm. |

* Các thông số có trong nodemailer

## Vận dụng:

1. Khởi tạo và thực hiện chạy chương trình gửi mail bằng nodejs:



# LAB 13: TẠO CLIENT REQUEST TRONG NODE.JS

## Mục tiêu:

* Biết về module Request.
* Khai báo và sử dụng module Request.

## Tóm tắt lí thuyết:

* module request là một module do lập trình viên phát triển dựa trên module http của Node.js nên để sử dụng được nó thì chúng ta cần phải cài đặt.
* Nếu như bạn muốn sử dụng RESful API thì module request cũng hỗ trợ bạn một cách đơn giản. Bạn chỉ cần trỏ đến các phương thức mà bạn muốn sử dụng với các phương thức mà module request hỗ trợ GET, POST, PUT, DELETE,….

## Vận dụng:

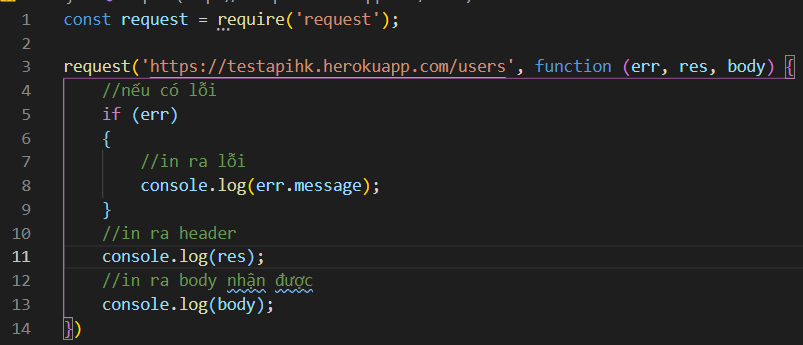
1. Cài đặt và sử dụng Request.
   * + Sử dụng npm cài đặt module Request.



* + - Khai báo module trước khi sử dụng.

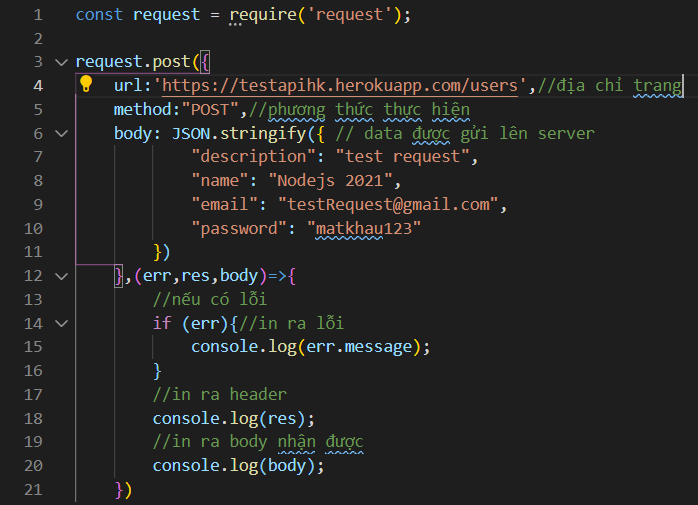


* + - Thực hiện request địa chỉ “ <https://testapihk.herokuapp.com/users> “



* + - Sau khi thực hiện chạy đoạn chương trình ta nhận được kết quả là body và các phương thức của header. Đoạn chương trình tương đương với method GET.

1. Cấu hình và sử dụng một số method với request.
   * + Thực hiện với method POST.



* + - Chạy đoạn chương trình dữ liệu data sẽ được gửi lên server và lưu lại.
    - Ngoài ra còn các phương thức khác (PUT, DELETE, PATCH,..) cũng thực hiện tương tự.

# LAB 14: MODULE HÓA TRONG NODE.JS

## Mục tiêu:

* Hiểu về module hóa.
* Biết module hóa trong nodejs.

## Tóm tắt lí thuyết:

* Module hóa hay nói cách khác là lập trình hướng module (cấu phần). Là một phương pháp lập trình mà khi đó chúng ta sẽ chia nhỏ dự án ra thành các module với mỗi module là một chức năng riêng,... Về phần lý thuyết bài này mình sẽ không nói nhiều mà chỉ tập trung ứng dụng nó vào Node.js thôi (mình sẽ viết phần lý thuyết này vào một bài khác).
* Cũng giống như các ngôn ngữ lập trình khác thì Node.js cũng hỗ trợ chúng ta chia dự án ra thành các module nhỏ để dễ cho việc bảo trì, nâng cấp,... Và trong Node.js thì việc modules hóa được thực thi qua cú pháp của ES6 (ECMA Script).
* trong Node.js chúng ta có thể exports một file ra thành module bằng cách sử dụng cú pháp sau:

“***module.exports = something;***”

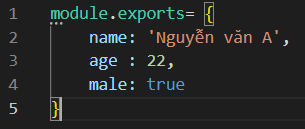
hoặc

“***exports = somthing;****”*

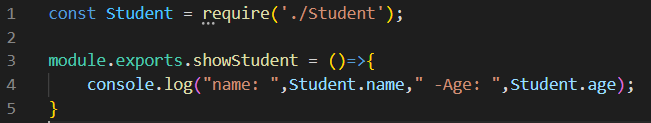
* + - Trong đó: ***Something***  là những phần cần được module hóa.

## Vận dụng:

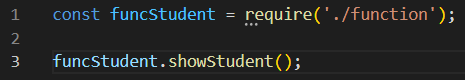
* + - 1. Thực hiện module hóa hàm, đối tượng.
    - Module hóa đối tượng student.



* + - Module hóa 1 chức năng và thực hiện khai báo module Student.



* + - Gọi module và thực thi chương trình.



Kết quả khi thực thi đoạn chương trình :



# LAB 15: NODE.JS VỚI MONGODB

## Mục tiêu:

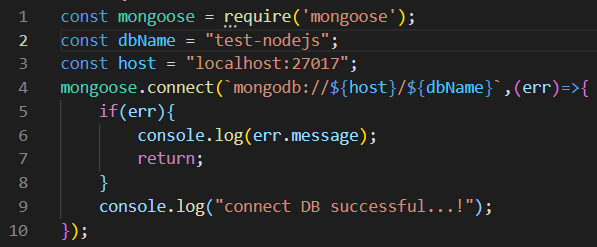
* Biết về mongoDB.
* Tìm hiểu về mongoose.
* Biết cấu hình và kết nối mongoDB trong nodejs.

## Tóm tắt lí thuyết:

* MongoDB là cơ sở dữ liệu dữ liệu được lưu trữ như một document, một tập của các cặp key-value. Bạn có thể định nghĩa nhiều database trong MongoDB và mỗi database có nhiều collections, những collections này đơn giản là tập của các documents được lưu trữ dạng cặp key-value..
* Mongoose là một thư việc rất phổ biến trong Node.js, thư viện này cung cấp một sự rành mạch, giải pháp schema-based để đối tượng hóa dữ liệu ứng dụng của bạn và bao gồm built-in type casting, validation, query, building, business logic và nhiều thứ khác.
* Trong Node.js chúng ta có thể thực hiện truy vấn cơ sở dữ liệu mongo và thao tác thêm xóa sửa các collections.

## Vận dụng:

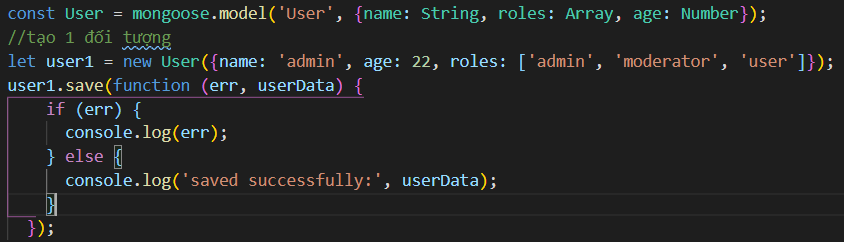
1. Khởi tạo và kết nối MongoDB (local):
   * + Tải và cài đặt mongDoB tại <https://www.mongodb.com/download>
     + Kết nối mongoDB trong nodejs với mongoose.

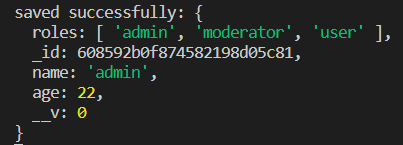


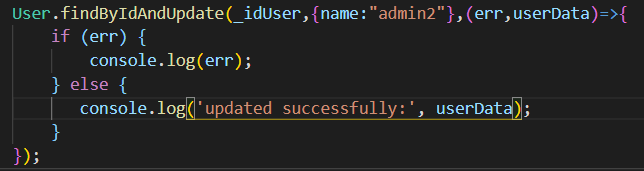
* + - Kết quả nhận được khi kết nối thành công:



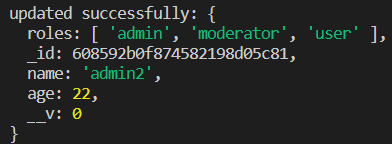
1. Xây dựng model user và thực hiện thao tác tạo thêm xóa sửa đối tượng.
   * + Khởi tạo model user và thêm 1 đối tượng



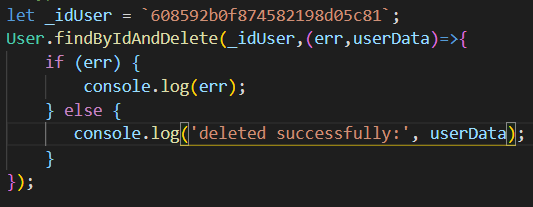
* + - Hàm **save()** dùng để thực hiện việc lưu vào CSDL.
    - Khi thực hiện đoạn chương trình và thêm thành công sẽ nhận được kết quả : 
    - Update một đối tượng



* + - Kết quả nhận được



* + - Delete một đối tượng



* + - Kết quả nhận được

