



**FPT POLYTECHNIC**



**THỰC HỌC – THỰC NGHIỆP**



**Conceive Design Implement Operate**

## **NODEJS & RESFUL WEB SERVICE**

### **CƠ BẢN VỀ NODEJS**

- ⊙ Hiểu được các khái niệm cơ bản về NodeJS
- ⊙ Thực hiện tạo project với NodeJS
- ⊙ Sử dụng được các hàm và đối tượng cơ bản trong NodeJS



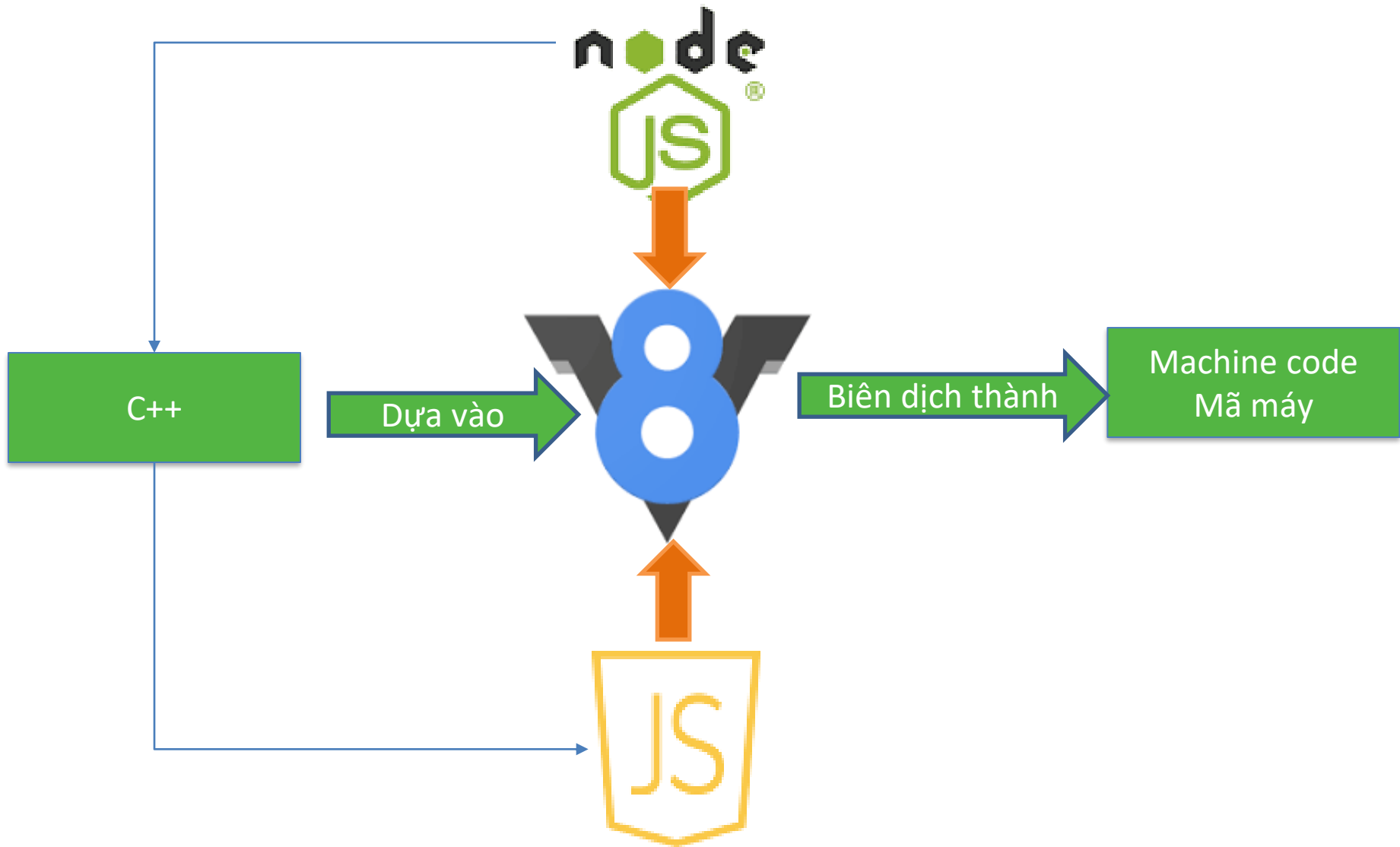
- 📖 NodeJS là gì?
- 📖 Công cụ
- 📖 Cài đặt NodeJS
- 📖 Tạo node server
- 📖 Router
- 📖 Request & Response





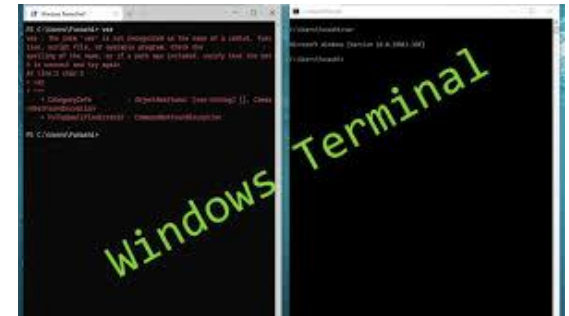
# PHẦN 1: TỔNG QUAN VỀ NODEJS

- ❑ [NodeJS](#) là một nền tảng được xây dựng trên “V8 Javascript engine” được viết bằng c++ và Javascript.
- ❑ Nền tảng này được phát triển bởi Ryan Lienhart Dahl vào năm 2009.



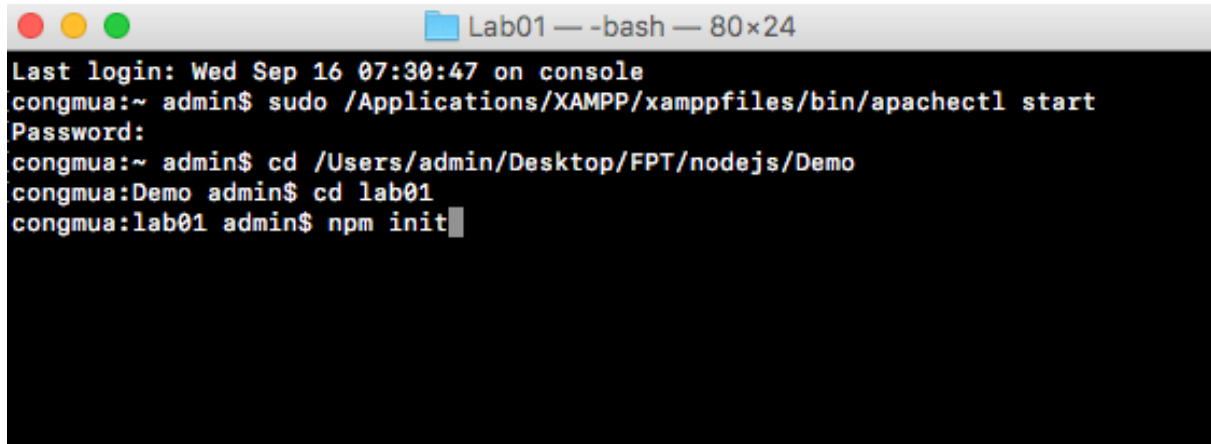


Visual Studio Code



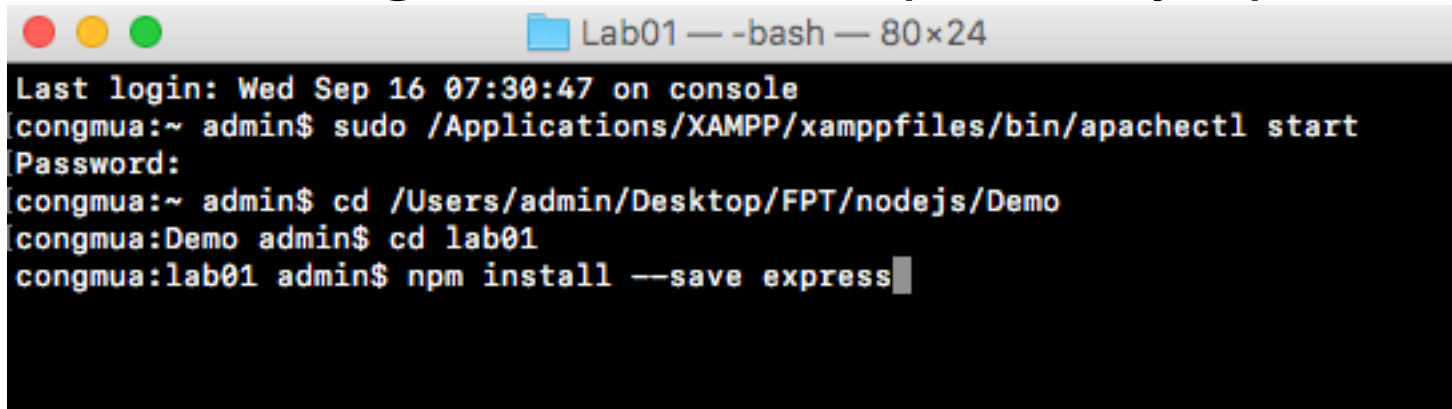
POSTMAN

## ❑ Khởi tạo project: npm init



```
Lab01 — -bash — 80x24
Last login: Wed Sep 16 07:30:47 on console
congmua:~ admin$ sudo /Applications/XAMPP/xamppfiles/bin/apachectl start
Password:
congmua:~ admin$ cd /Users/admin/Desktop/FPT/nodejs/Demo
congmua:Demo admin$ cd lab01
congmua:lab01 admin$ npm init
```

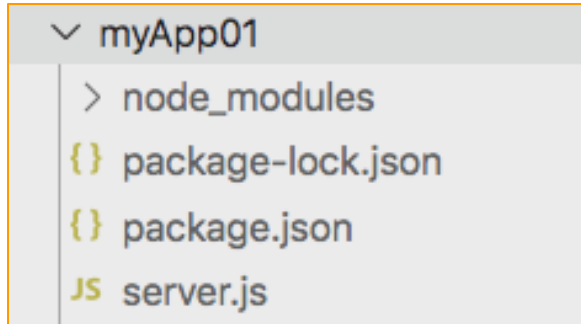
## ❑ Cài đặt các gói cần thiết: express,mysql, body-



```
Lab01 — -bash — 80x24
Last login: Wed Sep 16 07:30:47 on console
congmua:~ admin$ sudo /Applications/XAMPP/xamppfiles/bin/apachectl start
Password:
congmua:~ admin$ cd /Users/admin/Desktop/FPT/nodejs/Demo
congmua:Demo admin$ cd lab01
congmua:lab01 admin$ npm install --save express
```



## □ Tạo file server.js



```
{ } package.json ×
myApp01 > { } package.json > ...
1  {
2    "name": "myapp01",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "muatc",
10   "license": "ISC",
11   "dependencies": {
12     "express": "^4.17.1"
13   }
14 }
15
```

## Tạo server

```
Lab01 > JS serve.js > ...  
1  const http=require('http');  
2  server=http.createServer((req,res)=>{  
3    console.log(`ứng dụng đang chạy với port: 3000`);  
4  });  
5  server.listen(3000);
```

Để sử dụng module ta dùng hàm  
require()

Module là tập các chức năng mà ta  
có thể thêm vào ứng dụng khi cần

Module http giúp tạo server

Chạy server: node serve.js

TERMINAL	PROBLEMS	OUTPUT	DEBUG CONSOLE
congmu:Lab01 admin\$ node serve.js ứng dụng đang chạy với port: 3000			

Tạo server với express: Framework giúp ta xây dựng server hiệu quả hơn

```
JS server.js  X

myApp01 > JS server.js > ...
1  const express=require('express');
2  const app=express();
3  const port=3000;
4  app.listen(port,()=>{
5    |   console.log(`ứng dụng đang chạy với port: ${port}`);
6    | })
```

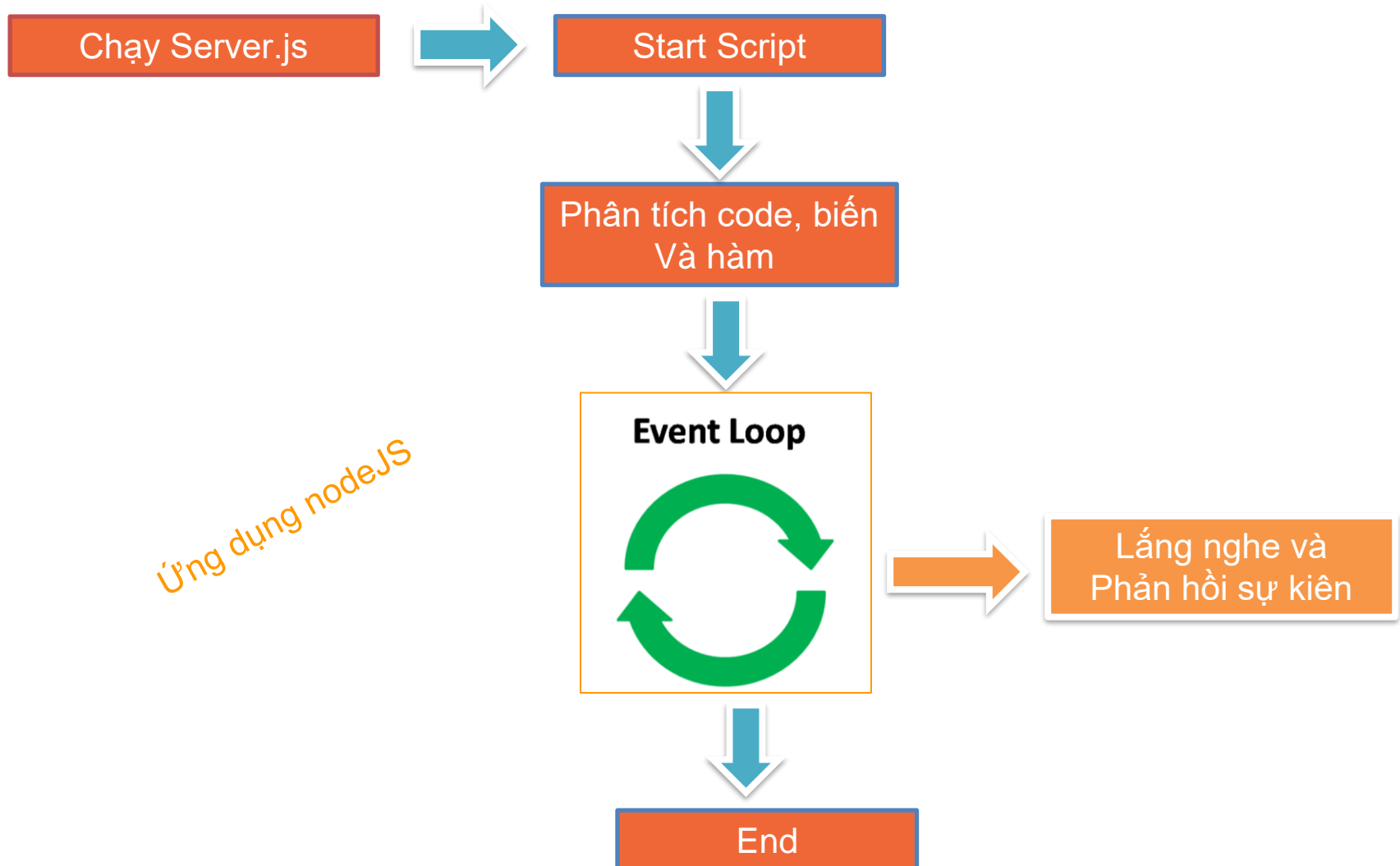
Chạy server: **npm start** hoặc **node server**

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  2:

congma:myApp01 admin$ npm start

> myapp01@1.0.0 start /Users/admin/Desktop/FPT/nodejs/Demo/myApp01
> node server.js

ứng dụng đang chạy với port: 3000
```



Một trong những tính năng của NodeJS là Không đồng bộ và điều khiển sự kiện



- Về cơ bản nó có nghĩa là một máy chủ dựa trên Node.js không bao giờ chờ đợi một API để trả về dữ liệu.
- Máy chủ chuyển sang API tiếp theo sau khi gọi nó
- Cơ chế thông báo của Sự kiện của Node.js giúp máy chủ nhận được phản hồi từ cuộc gọi API trước đó.

## Callback

### Blocking code

```
var fs = require("fs");  
var data = fs.readFileSync('input.txt');  
console.log(data.toString());  
console.log("Program Ended");
```

Program Ended

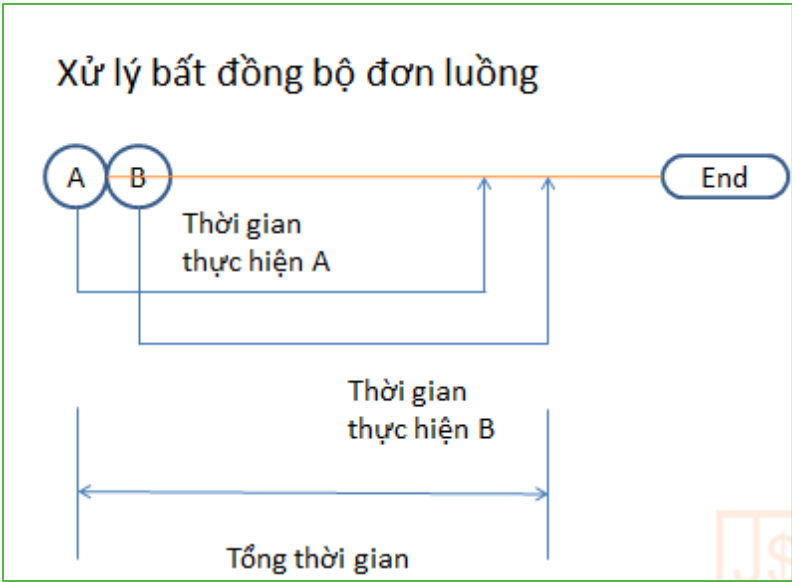
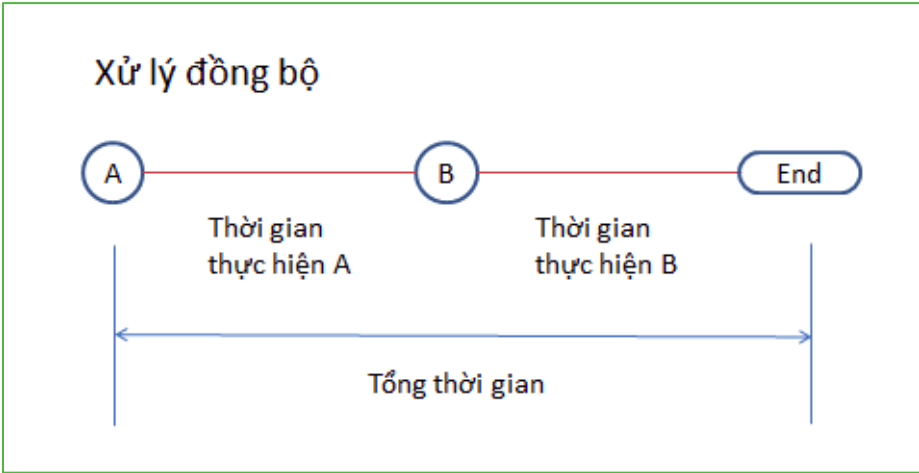
Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!

### Non-Blocking code

```
var fs = require("fs");  
fs.readFile('input.txt', function (err, data) {  
    if (err) return console.error(err);  
    console.log(data.toString());  
});  
console.log("Program Ended");
```

Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!  
Program Ended

## Callback



Vì Process B không cần chờ Process A thực hiện xong để B chạy, nên cần có hàm callback để khi B chạy xong thì A biết để đón bắt xử lý kết quả do B trả về

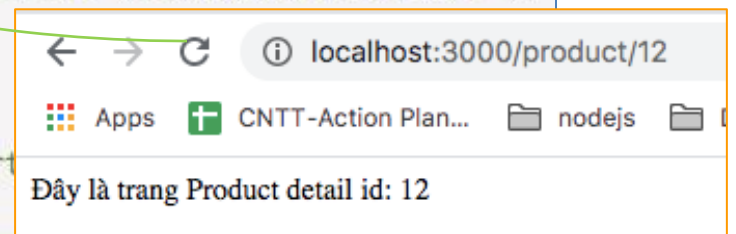
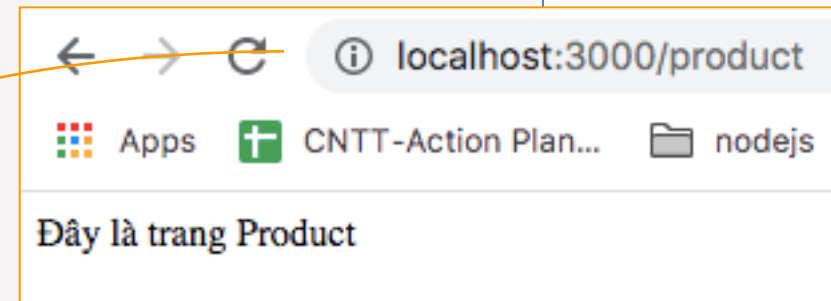
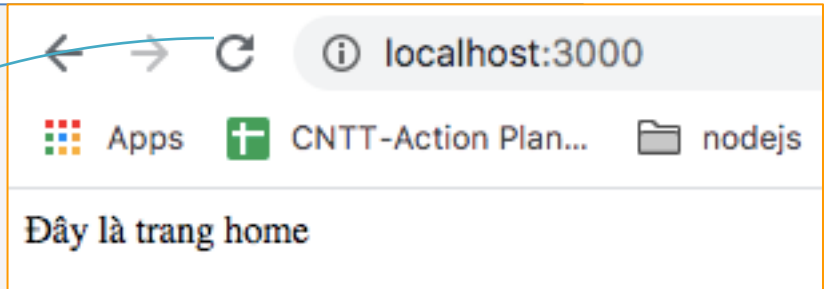
- ❑ **Router** là phương thức khai báo để đáp lại request từ client
- ❑ Định nghĩa URL cho trang web mà người dùng sẽ tương tác.



## Một số định nghĩa router thường dùng

```

JS server.js  X
myApp01 > JS server.js > ...
1  const express=require('express');
2  const app=express();
3  const port=3000;
4  //router
5  app.get('/', (req, res, next) => {
6    console.log('/ trang home');
7    res.send('<p>Đây là trang home</p>');
8  });
9
10 app.get('/product', (req, res, next) => {
11   console.log('product');
12   res.send('<p>Đây là trang Product</p>');
13 });
14
15 app.get('/product/:id', (req, res, next) => {
16   console.log('product detail');
17   res.send('<p>Đây là trang Product detail id: ${req.params.id}</p>');
18 });
19
20 app.listen(port,()=>{
21   console.log('ứng dụng đang chạy với port');
22 })
  
```

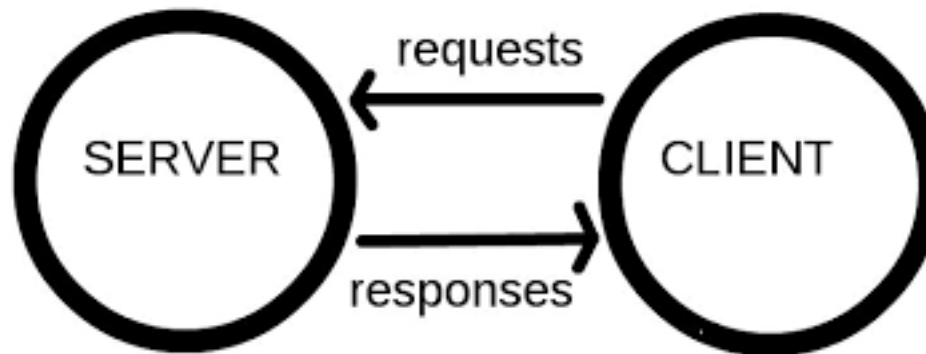


demo



## PHẦN 2: REQUEST & RESPONSE

- ❑ Là 2 đối tượng dùng để nhận(Request) và phản hồi(Response) các yêu cầu HTTP



- ❑ **Đối tượng Request** là tham số **req** trong callback function của route

```
// GET https://localhost:3000/product/123  
  
app.get('product/:id', (req, res) => {  
  console.log(req.params.id) // "123"  
})
```

- ❑ Giúp ta quản lý được yêu cầu từ phía client

- ❑ **Đối tượng Request** cung cấp các thuộc tính để nhận yêu cầu từ client
- ❑ Có 3 cách để chúng ta nhận dữ liệu từ người dùng
  - ❖ **Req.params**
  - ❖ **Req.query**
  - ❖ **Req.body**

## □ Req.params

```
// GET https://localhost:3000/product/123  
  
app.get('product/:id', (req, res) => {  
  console.log(req.params.id) // "123"  
})
```

## Req.query

<http://localhost:3000/product?id=123&name=Tao My>

```
app.get('/product', (req, res, next) => {  
  console.log(req.query);  
  res.send('<p>Đây là trang Product</p>');  
});
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

2: node

```
> node server.js
```

```
body-parser deprecated undefined extended: provide extended option server.js:6:20  
ứng dụng đang chạy với port: 3000  
{ id: '123', name: 'Tao My' }
```



## □ Req.body

```
const express=require('express');
var bodyParser = require('body-parser');
const app=express();
const port=3000;
// parsing application
app.use(bodyParser.urlencoded());
//form-urlencoded
//router
app.get('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text"
name="productName"><button type="submit">Add Product</button></form>');
});
app.post('/product', (req, res, next) => {
  console.log(req.body);
  res.send('<p>Đã thực hiện thêm Product</p>');
});
```

- ❑ **Đối tượng Request** là tham số **res** trong callback function của route

```
// GET https://localhost:3000/product  
  
app.get('/product', (req, res) => {  
  res.render('product.html', {name: 'Tao My'});  
});
```

- ❑ Giúp chúng ta một cách thức đơn giản để phản hồi các yêu cầu HTTP

❑ Một số phương thức của đối tượng response thường được dùng

❖ **res.send**

❖ **res.json**

❖ **res.status**

❖ **res.redirect**

❖ **res.render**

demo

- ☑ NodeJS là gì?
- ☑ Công cụ
- ☑ Cài đặt NodeJS
- ☑ Tạo node server
- ☑ Router
- ☑ Request & Response



thank  
you!