

# [FE-Vue-CLI env 설정] 개발 모드와 빌드 모드 구분으로 ems 호스트 정보 자동 설정 방법

## 요약

Vue-CLI는 커스텀 설정을 하지 않으면 빌드할 때 자동으로 process.env.NODE\_ENV에 값을 주입해 줍니다. 저는 코드 레벨에서 조건문을 사용해서 process.env.NODE\_ENV의 값에 따라서 개발 환경일 때와 빌드 환경에서 호스트를 설정해 주는 방법으로 코드 레벨에서 host(ems ip)를 자동으로 설정되도록 했습니다.

## 문제 해결 코드

```
//  
let gHost = `http://${window.location.host}`;  
if (process.env.NODE_ENV === 'development') {  
  gHost = `http://192.168.0.88:27889`; //  
}  
  
//  
let gHost = `http://${window.location.host}`; //  
if (process.env.NODE_ENV === 'development') {  
  gHost = `http://192.168.0.88:27889`;  
}
```

이번 글에서는 node 환경으로 구성된 edgemaster 웹 코드 베이스가 빌드 모드와 개발 모드를 구분해서 ems와 통신하는 host(ems IP)를 자동으로 설정하는 방법과 이 문제를 해결하는 과정을 소개하고자 합니다.

그리고 대학생 때, 공학윤리 수업 시간에 배운 엔지니어링적 접근법을 바탕으로 3가지 단계로 이 글의 목차를 구성해서 작성해 보았습니다.



## 문제 인식 및 정의

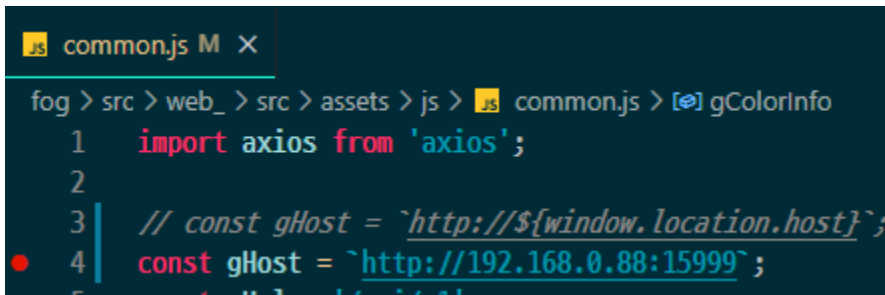
저는 매 번. 분기를 로컬로 받을 때마다, 아래의 코드가 반복적으로 제 손을 거치는 것이 문제라고 생각했습니다. (좀 늦게 깨달았습니다.)

```
- const gHost = `http://${window.location.host}`;
- // const gHost = `http://192.168.0.88:15999`;
```

(edgmaster/fog/src/web\_/src/assets/js/common.js 에서 코드를 확인할 수 있습니다.)

이 코드가 있으면 어떤 작업을 해줘야 하는 지를 예시로 보여드리겠습니다.

1. gitlab에서 분기를 로컬로 받고 빌드 모드로 사용되고 있는 gHost 변수를 손 수. 직접. ems ip와 port로 변경해 줘야 합니다. 아래처럼요.



```
common.js M X
fog > src > web_ > src > assets > js > common.js > gColorInfo
1 import axios from 'axios';
2
3 // const gHost = `http://${window.location.host}`;
4 const gHost = `http://192.168.0.88:15999`;
```

2. 로컬에서 작업 내용을 Gitlab에 업로드할 때 개발 모드로 켜둔 gHost 변수 때문에 common.js가 git 변경 사항에 항상 올라와 있습니다. 매 번 커밋을 할 때면 반영할 것도 아닌 common.js를 계속 보게 됩니다. 그리고, 실수로 커밋을 해버릴 수 있는 가능성도 있습니다.

Changes 1	History	fog#src#web_#src#assets#js#common.js	
<input checked="" type="checkbox"/> 1 changed file <input checked="" type="checkbox"/> fog#src#web_#src#assets#js#common.js			@@ -1,7 +1,7 @@
		1 1	import axios from 'axios';
		2 2	
		3	-const gHost = `http://\${window.location.host}`;
		4	-// const gHost = `http://192.168.0.88:15999`;
		3	+// const gHost = `http://\${window.location.host}`;
		4	+const gHost = `http://192.168.0.88:15999`;

문제의 원인을 정의하자면, 이 코드는 하드코딩이나 다른없는 방식을 사용하고 있어서 프로그램에 의한 자동 설정이 아니라 개발자가 수동으로 코드를 수정해야 하는 것이 문제입니다.

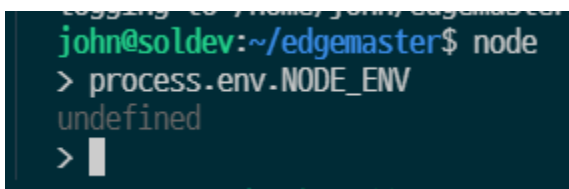
## 시행 착오 및 정보 분석

개발자의 피로도를 가중시키는 이 코드를 자동화시킬 방법을 찾아야 했습니다.

문제 해결 방법을 생각할 때 바로 떠오른 건, node의 process 객체에 빌드 환경을 설정하는 방법이었습니다.

88번 서버와 제 PC에서의 node에서 process객체의 env.NODE\_ENV 값을 직접 찾아봤습니다.

- 88서버 node



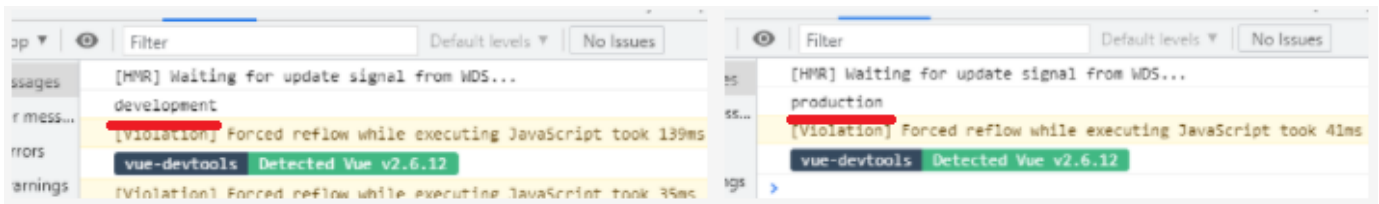
```
john@soldev:~/edgmaster$ node
> process.env.NODE_ENV
undefined
>
```

- 작업 PC node

```
MACH-NOT-10@DESKTOP-VKF4UH4 MINGW64 /c/abel/edgemaster/fog/src/web_ (690-chart-6months-bug)
$ node
Welcome to Node.js v14.15.4.
Type ".help" for more information.
> process.env.NODE_ENV
undefined
> |
```

설정이 없었습니다. 그런데, 코드 레벨에서 `console.log(process.env.NODE_ENV);`를 입력하고 브라우저 개발자 도구에서 확인하면 'production'과 'development'가 각각 출력되었습니다.

```
JS common.js M X
fog > src > web_ > src > assets > js > JS common.js > gEventLevelInfo > warning
1 import axios from 'axios';
2
3 console.log(process.env.NODE_ENV);
4 // const gHost = `http://${window.location.host}`;
5 const gHost = `http://192.168.0.88:15999`;
```



node에 설정되지 않은 상태들이 코드 레벨에서는 상태가 설정돼서 출력되고 있었습니다. node에도 설정되어 있지 않은 `process.env.NODE_ENV`의 상태가 어디서 세팅이 되고 있는지 알아봐야 했습니다.

Vue-CLI 공식 문서를 통해서 알아봤는데, Vue-CLI가 기본으로 제공하는 명령어를 통해서 코드를 빌드하면 `process.env.NODE_ENV`에 값을 자동으로 설정해주는 것을 알게 되었습니다.

# # Modes and Environment Variables

## Modes

Mode is an important concept in Vue CLI projects. By default, there are three modes:

- `development` is used by `vue-cli-service serve`
- `test` is used by `vue-cli-service test:unit`
- `production` is used by `vue-cli-service build` and `vue-cli-service test:e2e`

You can overwrite the default mode used for a command by passing the `--mode` option flag. For example, if you want to use development variables in the build command:

```
vue-cli-service build --mode development
```

When running `vue-cli-service`, environment variables are loaded from all **corresponding files**. If they don't contain a `NODE_ENV` variable, it will be set accordingly. For example, `NODE_ENV` will be set to `"production"` in production mode, `"test"` in test mode, and defaults to `"development"` otherwise.

edgemaster 웹 코드베이스는 Vue-CLI를 사용해서 자동 설정 방법으로 만들어진 프로젝트입니다. 저희가 번들링 설정을 커스터마이징 하지 않았고, Vue-CLI의 순수 스펙이 `process.env.NODE_ENV`를 설정해 주고있었습니다.

Vue-CLI를 통해서 막 생성한 프로젝트의 `package.json`파일을 열어보면, 공식 문서대로 `scripts` 객체에 자동으로 설정되어 있는 모드 별로 실행하는 명령어를 확인할 수 있습니다.



```
package.json 1 X
fog > src > web_ > package.json > {} dependencies
1  {
2    "name": "EdgeMaster",
3    "version": "0.1.0",
4    "private": true,
5    "scripts": {
6      "serve": "vue-cli-service serve --port 9001",
7      "build": "vue-cli-service build",
8      "lint": "vue-cli-service lint"
9    },
10   }
```

Vue-CLI 명령어가 실행되면 발생하는 일들의 과정을 간단히 정리해 보았습니다.

빌드 모드 : `$npm run build -> 빌드(process.env.NODE_ENV = 'production') -> 번들링 된 dist 폴더가 생성됨.`

개발 모드 : \$npm run serve -> 빌드(process.env.NODE\_ENV = 'development') -> dev-server로 코드가 실행됨.

이제야 알게 되었지만, Vue-CLI 덕분에 edgemaster는 빌드 모드와 개발 모드를 이미 구분해서 사용하고 있었던 겁니다.



## 문제 해결 방법

Vue-CLI의 기본 스펙으로 자동으로 모드가 설정되는 것을 알았고, 제가 직접 node에 환경변수를 설정하는 작업은 필요가 없었습니다. 코드 레벨에서 Vue-CLI가 빌드를 하면서 설정해 준 process.env.NODE\_ENV를 조건문에 사용해서 host가 자동 설정이 되도록 프로그래밍해서 모드에 따라서 자동으로 host가 설정되도록 문제를 해결했습니다.

### 코드

```
//
let gHost = `http://${window.location.host}`;
if (process.env.NODE_ENV === 'development') {
  gHost = `http://192.168.0.88:27889`; //
}

//
let gHost = `http://${window.location.host}`; //
if (process.env.NODE_ENV === 'development') {
  gHost = `http://192.168.0.88:27889`;
}
```

글이 재미있고 유익하셨다면, 밑에 [Like](#) 한 번씩 눌러주세요.

끝!