

2장. Vue-CLI



목 차

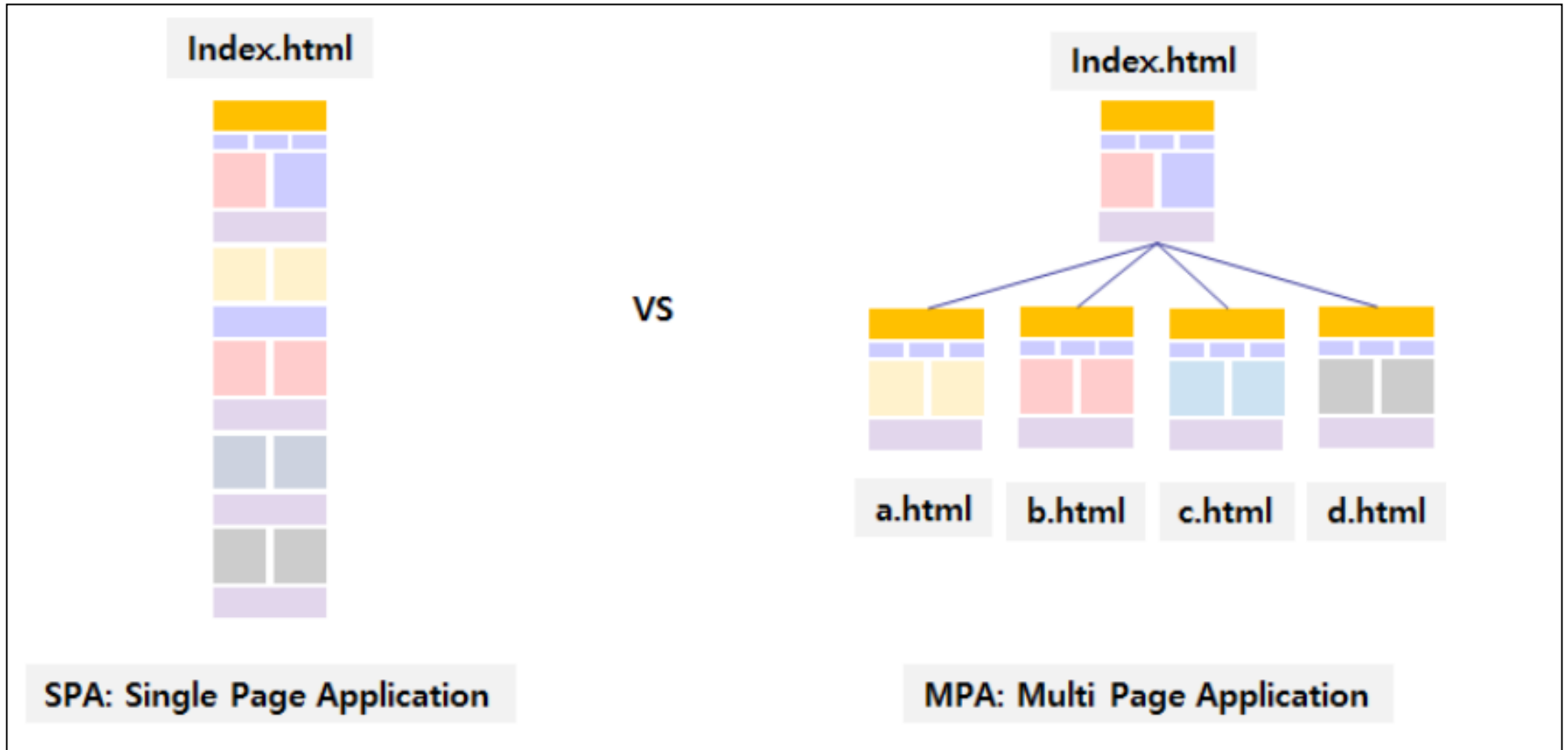
Vue-CLI 개요

SPA vs MPA

SFC

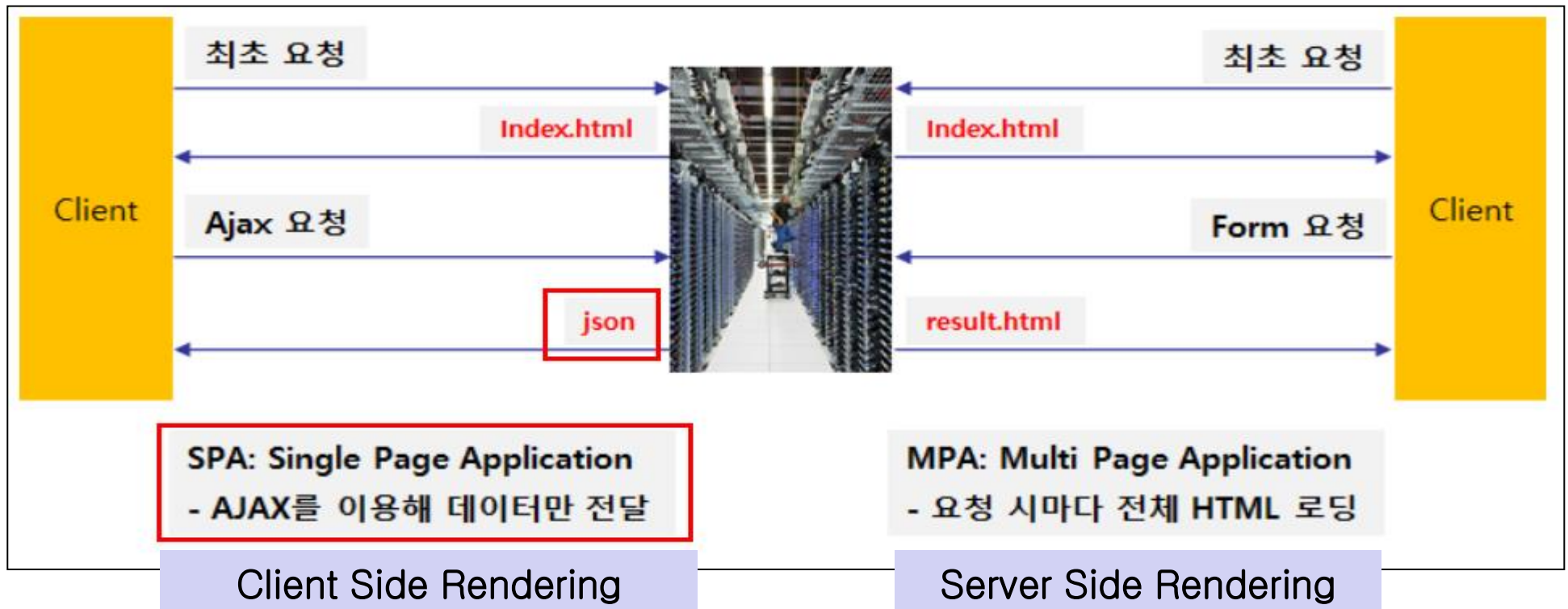
1. SPA vs MPA

웹 어플리케이션의 UI 화면을 구현할 때 사용 가능한 형태는 SPA와 MPA가 있다.



1. SPA vs MPA

SPA와 MPA의 동작 방식의 가장 큰 차이점은 일반적인 Form 전송이나 Ajax 동작이냐로 나눌 수 있다.



SPA는 Single Page Application으로 Client Side Rendering을 추구한다. 즉 UI화면과 관련된 리소스를 처음 요청시 서버로부터 몽땅 받아낸 후 클라이언트에서 모든 HTML/CSS/JS를 가지고 있다. 이후 Ajax 통신을 통해 변경하고자 하는 데이터만 받아오게 된다.

장점

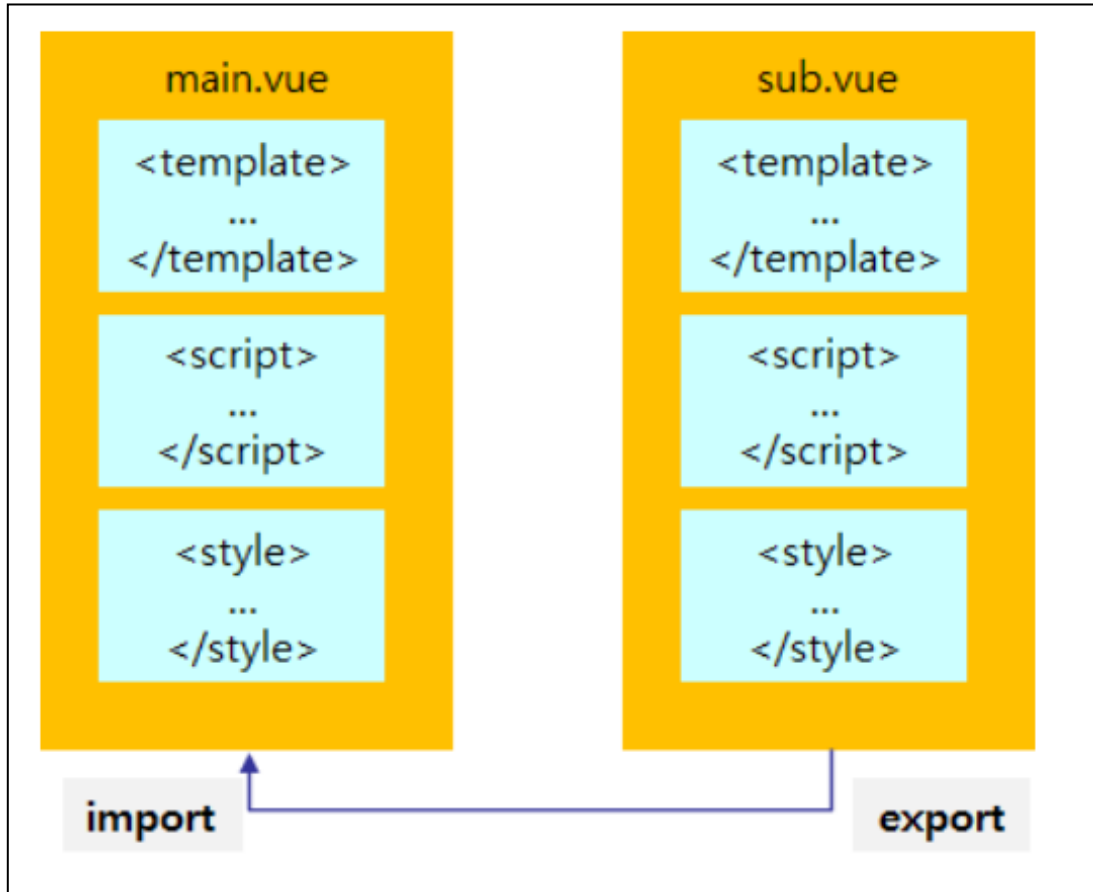
- SPA는 사용 중 리소스 로딩이 없기 때문에 부드럽게 화면 전환이 이루어진다.
- 서버 입장에서는 템플릿(화면)을 만드는 연산이 클라이언트로 분산되기 때문에 부담이 줄어든다.
- 컴포넌트별로 개발하기 때문에 생산성이 향상된다.
- 모바일 앱에서도 동일한 패턴의 Rest API 사용이 가능하다.

단점

- URL이 변경되지 않기 때문에 검색엔진의 색인화가 어렵다.
- 초기 구동 비용이 MPA 대비 상대적으로 비싸다.

3. Vue와 SFC (Single File Component)

SFC는 단어 뜻 그대로 파일 하나가 하나의 컴포넌트가 된다는 개념이다.
Vue는 이를 위해서 **.vue** 확장자의 파일을 이용하는데 이것은 다음과 같은 구조로 관리된다.

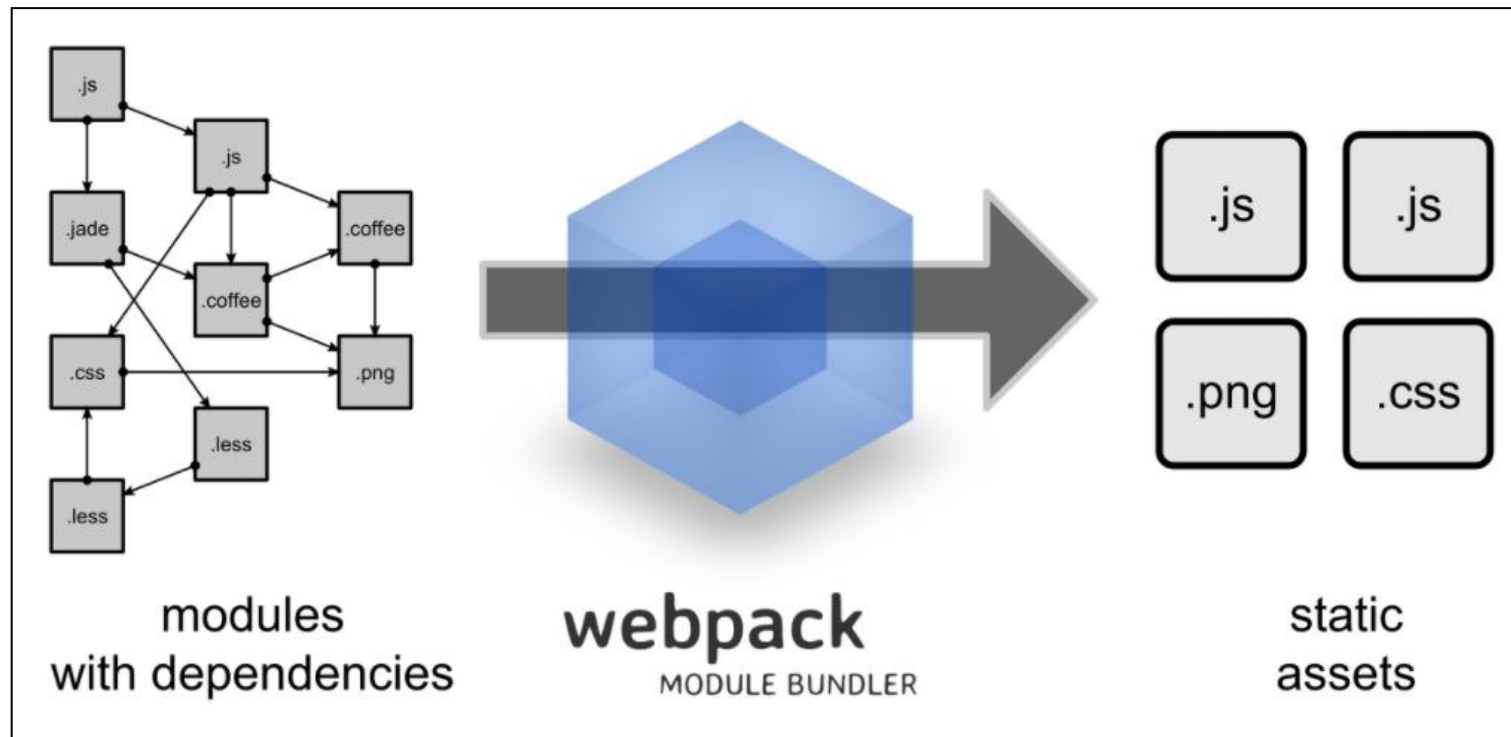


<template> 역할은 HTML 작성이고 <style>은 CSS, <script>는 JavaScript 영역이다. 즉, 하나의 컴포넌트를 만들때 필요한 요소들이 하나의 파일에 모이는 것이다. 이렇게 만들어진 모듈들은 import와 export를 이용해서 서로 간에 참조가 가능해진다.

4. vue를 위한 전처리 작업

일단 .vue로 개발된 파일들은 WebPack과 같은 모듈 빌더 툴을 이용해서 우리에게 익숙한 html 형태로 변환된다.

단순히 파일이 변환되는 것이 아니라 Babel과 같은 시스템을 이용해서 ECMA6등 높은 버전의 스크립트를 ECMA5로 다운 그레이드 시켜주기 때문에 하위 브라우저 호환성도 해결될 수 있다.



복잡한 WebPack을 보다 편리하게 사용할 수 있도록 도와주는 도구가 CLI가 된다.

5. Vue-CLI

<https://cli.vuejs.org/guide/installation.html>

1) Vue-CLI 설치

To install the new package, use one of those commands:

```
npm install -g @vue/cli  
# OR  
yarn global add @vue/cli
```

```
vue --version
```

```
C:\WINDOWS\system32>npm install -g @vue/cli  
npm WARN deprecated @hapi/topo@3.1.6: This version
```

```
C:\WINDOWS\system32>vue --version  
@vue/cli 4.5.15
```

도움말

```
C:\WINDOWS\system32>vue --help
Usage: vue <command> [options]

Options:
  -V, --version          output the version number
  -h, --help             output usage information

Commands:
  create [options] <app-name>      create a new project powered by vue-cli-service
  add [options] <plugin> [pluginOptions]  install a plugin and invoke its generator in an already created project
  invoke [options] <plugin> [pluginOptions]  invoke the generator of a plugin in an already created project
  inspect [options] [paths...]  inspect the webpack config in a project with vue-cli-service
  serve [options] [entry]        serve a .js or .vue file in development mode with zero config
  build [options] [entry]        build a .js or .vue file in production mode with zero config
  ui [options]                  start and open the vue-cli ui
  init [options] <template> <app-name>  generate a project from a remote template (legacy API, requires @vue/cli-init)
  config [options] [value]          inspect and modify the config
  outdated [options]               (experimental) check for outdated vue cli service / plugins
  upgrade [options] [plugin-name]  (experimental) upgrade vue cli service / plugins
  migrate [options] [plugin-name]  (experimental) run migrator for an already-installed cli plugin
  info                             print debugging information about your environment

Run vue <command> --help for detailed usage of given command.
```

상단에 있는 create와 add 명령이 가장 많이 사용되는 명령으로서 각각 프로젝트를 생성하거나 라이브러리를 추가할 때 사용된다. 마지막으로 command별로 도움말을 보려면 `vue <command> --help` 로 하면 된다.

5. Vue-CLI

<https://cli.vuejs.org/guide/creating-a-project.html>



2) Project 생성

```
vue create hello-world
```

```
vue create <프로젝트명>
```

```
c:\wvue>vue create my-app

Vue CLI v4.5.15
? Please pick a preset: (Use arrow keys)
> Default ([Vue 2] babel, eslint)
  Default (Vue 3) ([Vue 3] babel, eslint)
  Manually select features
```

 Successfully created project **my-app**.
 Get started with the following commands:

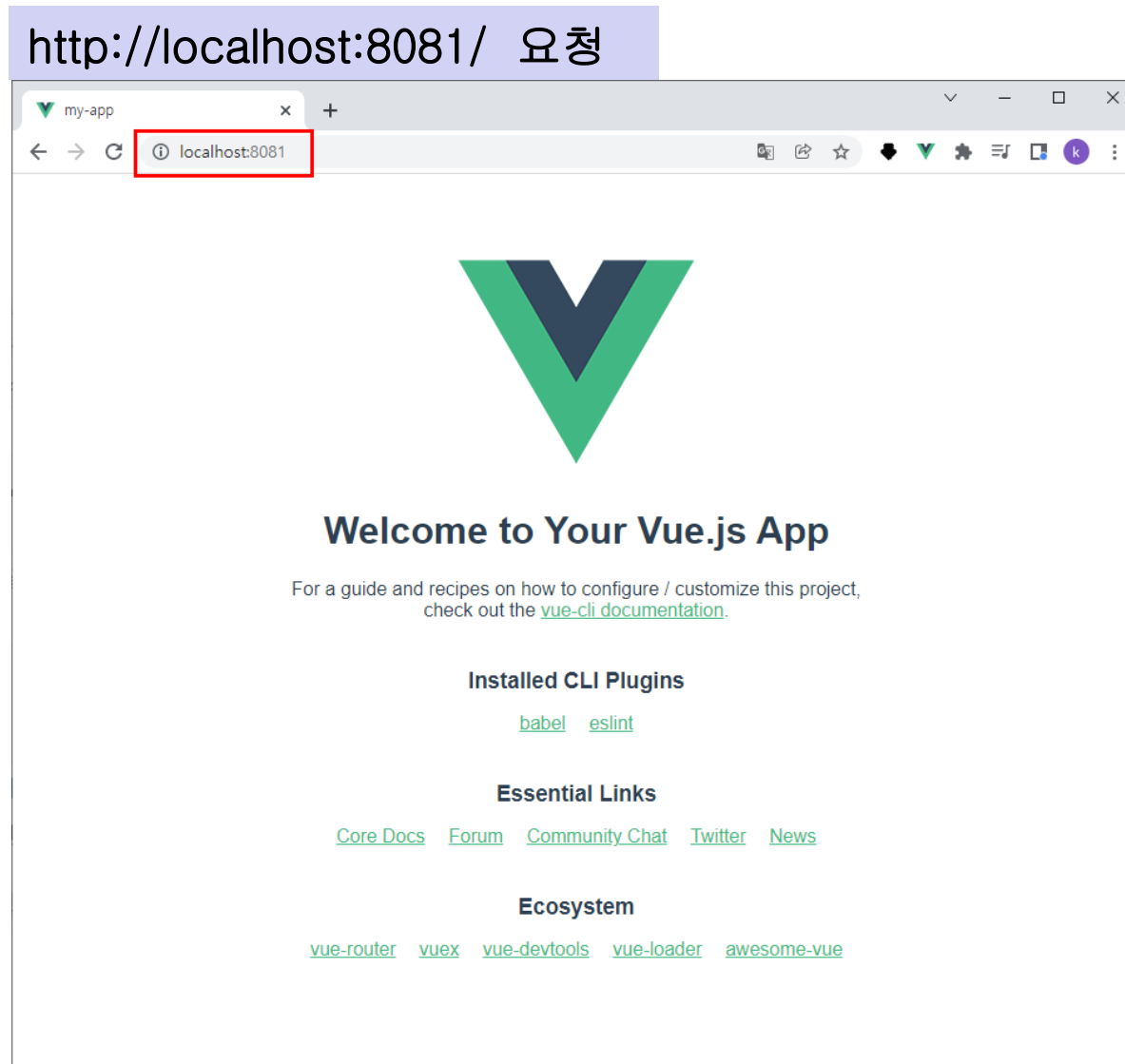
```
$ cd my-app
$ npm run serve
```

```
c:\wvue>cd my-app
```

```
c:\wvue\my-app>npm run serve
```

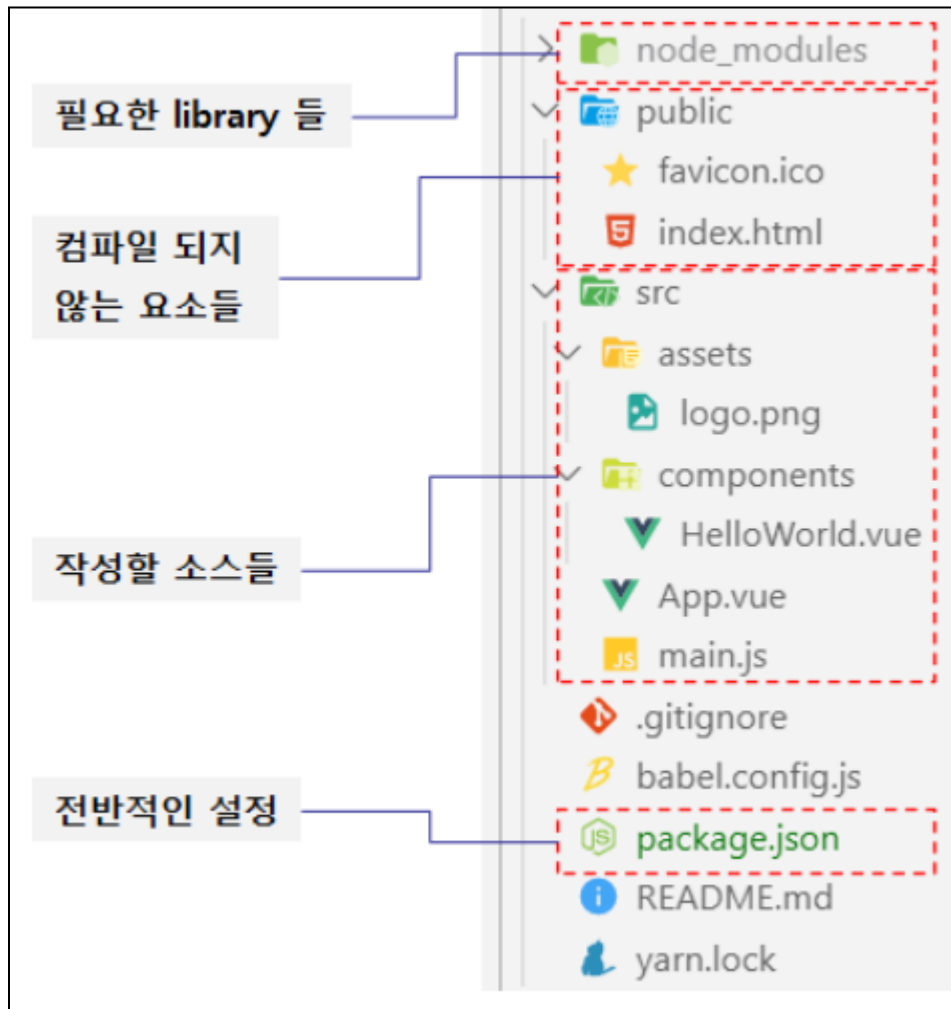
App running at:
- Local: <http://localhost:8081/>
- Network: <http://192.168.0.10:8081/>

5. Vue-CLI



6. 생성된 프로젝트 구조

Vue.js



node_modules

앱 개발과 배포에 필요한 npm 패키지들이 저장된다.

public

배포 버전을 빌드할 때 필요한 파일들이다. 웹팩을 이용해서 이 파일을 로드한 뒤 설정을 추가해서 빌드 버전이 완성된다.

src

컴포넌트들을 구성하는 영역으로 Vue가 컴파일 하는 대상이다. `assets`는 image와 같은 자원들이 저장되는 곳으로 component에서 이 곳의 파일들을 사용하는 경우 자동으로 배포된다.

package.json

프로젝트에 대한 전반적인 설정이 저장된다.

7. package.json

프로젝트의 설정을 저장하는 package.json 파일 내용은 다음과 같다.

```
"scripts": {  
  "serve": "vue-cli-service serve --open",  
  "build": "vue-cli-service build",  
  "lint": "vue-cli-service lint"  
},
```

Project 관련 유틸리티 script들
serve: 프로젝트를 서버에서 실행시킴
build: 컴파일 → 배포 처리
lint: eslint로 문법 체크

```
"dependencies": {  
  "core-js": "^3.6.5",  
  "vue": "^2.6.11"  
},
```

프로젝트 동작에 필요한 라이브러리 정보
npm install <library>

```
"devDependencies": {  
  "@vue/cli-plugin-babel": "~4.4.0",  
  ...  
  "vue-template-compiler": "^2.6.11"  
},
```

프로젝트 개발에 필요한 라이브러리 정보
npm install -D <library>

```
"eslintConfig": {  
  ...  
  "rules": {  
    "no-console": "off"  
  }  
}
```

eslint 정보

- 관행적 위험에 대한 오류 처리
- 힌트 제공
- 콘솔은 그냥 쓰자..

7. package.json

nodejs를 이용해서 개발을 하다보면 node_modules 경로에 다운로드 받은 많은 라이브러리들이 저장된다.

개발 배포시 이것까지 배포하기에는 많은 용량이기 때문에 실제 배포할 때는 node_modules의 내용을 제외하고 배포하게 된다.

배포후 그것을 사용하려면 package.json이 있는 폴더에서 **npm install** 명령을 이용하면 관련 dependency가 한꺼번에 다운로드 된다.

8. src/main.js

Vue-CLI로 작성된 application의 분석은 main.js에서 시작된다. main.js에서 최종적으로 Vue컴포넌트들을 조합해서 배포시 index.html에 표시된다.

```
1 import Vue from "vue";
2 import App from "./App.vue";
3
4 Vue.config.productionTip = true;
5
6 new Vue({
7   render: (h) => h(App),
8 }).$mount("#app");
```

1번 라인은 핵심 객체인 Vue를 impor한다.

2번 라인은 개발하려는 SFC의 첫 화면인 App.vue를 impor한다.

Vue.config.productionTip은 Vue앱이 처음 실행될 때 나오는 경고문을 출력할 것인지 물어보는 내용이다. 상용인 경우는 false로 지정한다.

6번 라인부터는 id가 app인 html태그에 App.vue의 내용을 표시하는 문장이다.

다음은 최종 화면에 보여줄 index.html 파일의 templat이다. <div id='app'> 영역에 컴파일된 vue컴포넌트가 삽입된다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <link rel="icon" href="<%= BASE_URL %>favicon.ico" />
    <title><%= htmlWebpackPlugin.options.title %></title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't work
      properly without JavaScript enabled. Please enable it to continue.</strong>
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

중간에 `<%=BASE_URL%>`는 웹팩의 내용을 참조하는 것으로서 변경이 필요하다면 프로젝트 root경로에 `vue.config.js` 파일을 만들고 내용을 수정할 수 있다.

README.md

```
23   ### Customize configuration  
24   See [Configuration Reference](https://cli.vuejs.org/config/).  
25
```

변경 예

```
module.exports = {  
  publicPath: "/ssafy_vue", // <%=BASE_URL%>  
  assetsDir: "assets",  
  pages: {  
    app: {  
      entry: "src/main.js",  
      template: "public/index.html",  
      filename: "index.html",  
      title: "SSAFY", // <%=htmlWebpackPlugin.options.title%>  
    },  
  },  
};
```


.vue 파일의 구성요소

SFC는 화면을 구성하는 3가지 요소들을 하나로 묶어주는 것으로서 이를 위해 .vue는 3가지 태그를 사용한다.

<template> 요소

화면을 구성하는 html부분으로 반드시 하나의 root 태그를 가져야 한다. 기존 작성과 차이점은 하나의 template만 존재하기 때문에 별도의 id값을 지정하지 않는다.

<script> 요소

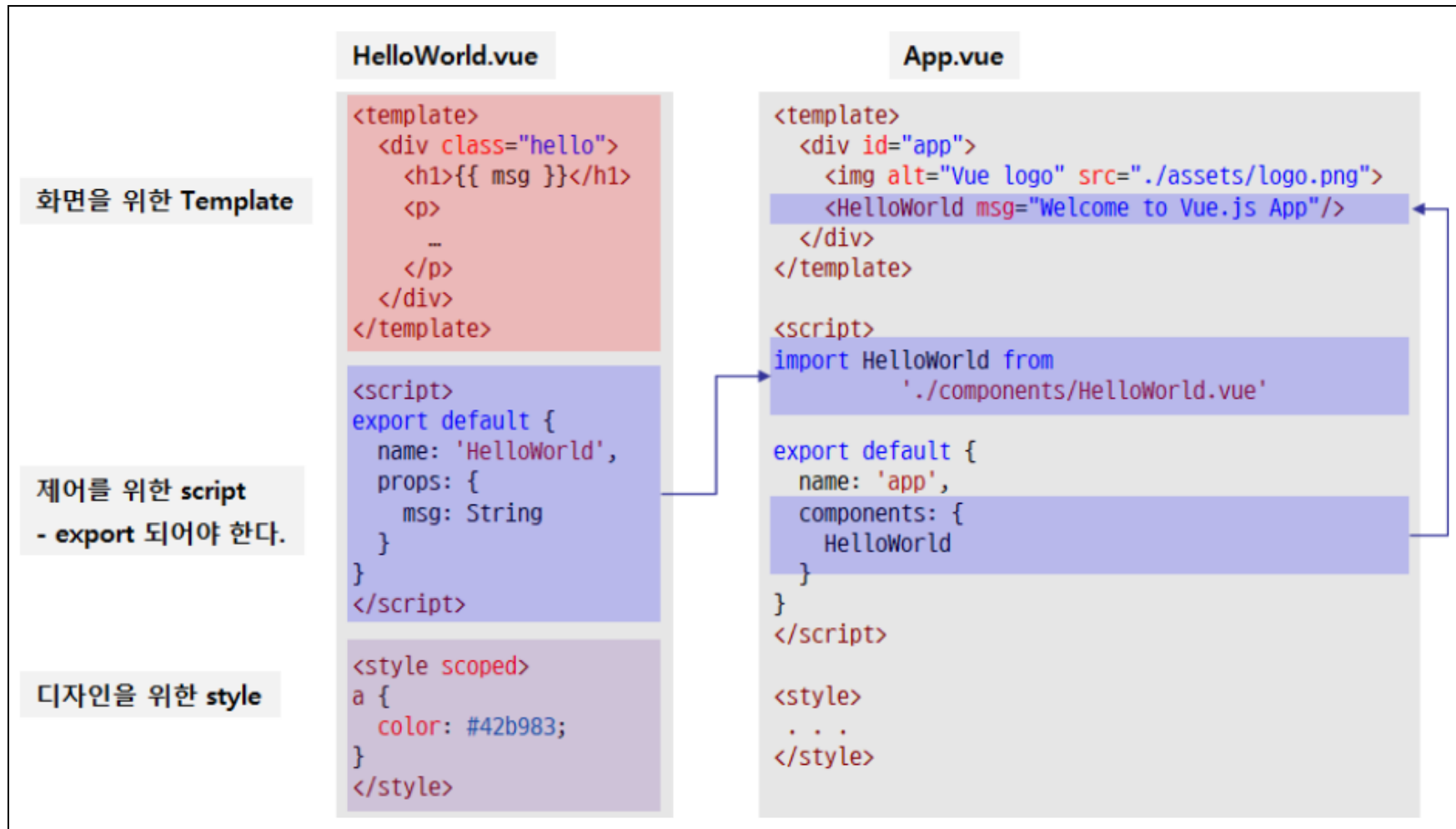
컴포넌트를 만들어서 export 해주는 역할이다. 객체는 하나만 만들고 default로 export 한다.

<style> 요소

스타일을 담당하는 CSS 코드가 들어간다. 기본적으로 전역 레벨이 되어서 다른 컴포넌트에 영향을 줄 수 있기 때문에 현재 컴포넌트에게만 국한시킬 경우 scoped 속성을 지정한다.

11. 제공된 .vue의 분석

기본적으로 제공된 .vue 파일을 살펴보면 앞서 설명했듯이 main.js에서 App.vue를 import하고 App.vue는 다시 HelloWorld.vue를 import한다.



<template>

<template>은 main.js의 #app 영역에 표현되는 html코드이다. 하위 태그로는 HelloWorld가 선언되어 있는데 바로 하위 컴포넌트이다. <HelloWorld>에 msg 속성은 HelloWorld의 props 속성에 값을 할당하는 것이다.

<script>

컴포넌트를 만들어서 export 하는 javascript 작성 부분이다. 다른 컴포넌트인 HelloWorld.vue를 import 하고 있으며 하위 컴포넌트를 등록하기 위해서 components 속성을 사용한다.

<style>

스타일을 작성하는 부분으로 로컬에서만 사용하는 경우에는 scoped 속성을 지정하고 전역에서 사용하려면 생략한다.

12. BookList 샘플



App.vue

BookList.vue

13. 실습문제

