가이드 문서  
Vue 코딩 가이드

작성자 : 최세훈

short line

목차

[1. 파일 공통 요건 3](#_Toc124776582)

[1.1 파일 인코딩은 UTF-8 3](#_Toc124776583)

[2. 이름 (Naming) 3](#_Toc124776584)

[2.1 역할별 파일명 3](#_Toc124776585)

[2.2 한국어 발음대로의 표기 지양 4](#_Toc124776586)

[2.3 패키지 이름은 소문자로 구성 4](#_Toc124776587)

[2.4 클래스/인터페이스 이름에 파스칼표기법 적용 5](#_Toc124776588)

[2.5 메서드/함수 이름에 카멜표기법 적용 5](#_Toc124776589)

[2.6 메서드/함수 이름은 동사/전치사로 시작 5](#_Toc124776590)

[2.7 template내의 commponet는 케밥표기법 적용 6](#_Toc124776591)

[2.8 상수는 대문자와 언더스코어로 구성 6](#_Toc124776592)

[2.9 변수에 소문자 카멜표기법 적용 6](#_Toc124776593)

[2.10 임시 변수 외에는 1 글자 이름 사용 금지 7](#_Toc124776594)

[3. 선언 (Declaration) 7](#_Toc124776595)

[3.1 한 줄에 한 문장 7](#_Toc124776596)

[3.2 let 보단 const 사용 8](#_Toc124776597)

[4. 들여쓰기 (Indentation) 8](#_Toc124776598)

[4.1 탭의 크기는 2개의 스페이스 8](#_Toc124776599)

[4.2 블럭 들여쓰기 8](#_Toc124776600)

[5. 중괄호 (Braces) 8](#_Toc124776601)

[5.1 K&R 스타일로 중괄호 선언 8](#_Toc124776602)

[5.2 닫는 중괄호와 같은 줄에 else, catch, finally, while 선언 9](#_Toc124776603)

[5.3 빈 블럭에 새줄 없이 중괄호 닫기 허용 10](#_Toc124776604)

[5.4 조건/반복문에 중괄호 필수 사용 10](#_Toc124776605)

[6. 공백 (Whitespace) 11](#_Toc124776606)

[6.1 중괄호의 시작 전, 종료 후에 공백 삽입 11](#_Toc124776607)

[6.2 제어문 키워드와 여는 소괄호 사이에 공백 삽입 12](#_Toc124776608)

[6.3 식별자와 여는 소괄호 사이에 공백 미삽입 12](#_Toc124776609)

[6.4 콤마/구분자 세미콜론의 뒤에만 공백 삽입 13](#_Toc124776610)

[6.5 콜론의 앞 뒤에 공백 삽입 13](#_Toc124776611)

[6.6 이항/삼항 연산자의 앞 뒤에 공백 삽입 13](#_Toc124776612)

[6.7 단항 연산자와 연산 대상 사이에 공백을 미삽입 14](#_Toc124776613)

[6.8 주석문 기호 전후의 공백 삽입 14](#_Toc124776614)

[7. 기타 15](#_Toc124776615)

[7.1 표기법 15](#_Toc124776616)

[8. 참고자료 15](#_Toc124776617)

해당 가이드 문서에서 제시하는 코딩 가이드를 준수하여 소프트웨어 개발 시에 다수의 개발자들이 상호간의 소스코드에 대한 가독성 및 이해도를 높이고, 해당 표준에 따라 개발함으로써 프로젝트 품질의 일관성을 유지하여 프로젝트 완료 이후의 원활한 시스템 유지보수를 지원할 수 있도록 하는데 있다.

# 파일 공통 요건

## 파일 인코딩은 UTF-8

모든 소스, 텍스트 문서 파일의 인코딩은 UTF-8로 통일한다

# 이름 (Naming)

## 역할별 파일명

a) compoment의 파일명은 파스칼 케이스([표기법](#_표기법))를 사용한다  
*- 나쁜 예*

|  |
| --- |
| components/  |- mycomponent1.vue |- myComponent2.vue |

*- 좋은 예*

|  |
| --- |
| components/  |- MyComponent1.vue |- MyComponent2.vue |

b) script (javascript/typescript) 의 파일명은 카멜케이스([표기법](#_표기법)) 사용한다  
파일명 포맷은 {파일명}.{역할}.ts 을 사용한다. 역할은 interface, model, enum, api, module 과 같다. // TODO .{역할}.ts 를 사용이 맞을까..?  
*- 나쁜 예*

|  |
| --- |
| service  |- my-visit.api.ts |-model/ |-- MyVisit1.interface.ts |-- MyVisit2.interface.ts |

*- 좋은 예*

|  |
| --- |
| service  |- myVisit.api.ts |-model/ |-- myVisit1.interface.ts |-- myVisit2.interface.ts |

## 한국어 발음대로의 표기 지양

식별자의 이름을 한글 발음을 영어로 옮겨서 표기하지 않는다. 한국어 고유명사는 예외이다.  
- 나쁜 예 : moohyungJasan (무형자산)  
- 좋은 예 : intangibleAssets (무형자산)

## 패키지 이름은 케밥표기법 사용

패키지 이름은 케밥표기법을 사용하여 작성한다. 단어별 구문을 위해 언더스코어(\_)나 대문자를 섞지 않는다.  
*- 나쁜 예*

|  |
| --- |
| import { HomeTopNotice } from '@/service/Myhome/model/homeTopNotice.interface';  import { HomeTopNotice } from '@/service/myHome/model/homeTopNotice.interface'; |

*- 좋은 예*

|  |
| --- |
| import { HomeTopNotice } from '@/service/my-home/model/homeTopNotice.interface'; |

## 클래스/인터페이스 이름에 파스칼표기법 적용

클래스 이름은 단어의 첫 글자를 대문자로 시작하는 파스칼표기법을 사용한다.   
*- 나쁜 예*

|  |
| --- |
| export class reservation  export interface accessToken |

*- 좋은 예*

|  |
| --- |
| export class Reservation  export interface AccessToken |

## 메서드/함수 이름에 카멜표기법 적용

메서드/함수의 이름에는 첫 번째 단어를 소문자로 작성하고, 이어지는 단어의 첫 글자를 대문자로 작성하는 카멜표기법를 사용한다.

## 메서드/함수 이름은 동사/전치사로 시작

메서드/함수 명은 기본적으로는 동사로 시작한다.  
*- 좋은 예*

|  |
| --- |
| 동사사용 : renderHtml()  전환메서드의 전치사 : obj.toString() |

## template내의 commponet는 케밥표기법 적용

< template></template> 내의 commponet를 사용할때, single componet, dom template 모두 케밥표기법을 사용한다.

*- 나쁜 예*

|  |
| --- |
| <!-- In Single-File Components, string templates, and JSX -->  <MyComponent></MyComponent>  <!-- In DOM templates -->  <MyComponent/> |

*- 좋은 예*

|  |
| --- |
| <!-- In Single-File Components, string templates, and JSX -->  <my-component/>  <!-- In DOM templates -->  <my-component></my-component> |

## 상수는 대문자와 언더스코어로 구성

상수는 대문자 스네이크 표기법을 사용한다.  
*- 좋은 예*

|  |
| --- |
| BASE\_URL  VUE\_APP\_API\_PATH |

## 변수에 소문자 카멜표기법 적용

상수가 아닌 멤버변수/지역변수/메서드/함수 파라미터에는 카멜표기법을 사용한다.  
*- 나쁜 예*

|  |
| --- |
| const Authorized: boolean;  let AccessToken: string; |

*- 좋은 예*

|  |
| --- |
| const authorized: boolean;  let accessToken: string; |

## 임시 변수 외에는 1 글자 이름 사용 금지

메서드/함수 블럭 범위 이상의 생명 주기를 가지는 변수에는 1글자로 된 이름을 쓰지 않는다. 반복문의 인덱스나 arrow 표현식의 파라미터 등 짧은 범위의 임시 변수에는 관례적으로 1글자 변수명을 사용할 수 있다.  
*- 나쁜 예*

|  |
| --- |
| const t: string; |

*- 좋은 예*

|  |
| --- |
| const token: string; |

# 선언 (Declaration)

필드, 메서드, 함수, 변수값, import문 등의 소스 구성요소를 선언할 때 고려해야할 규칙이다

## 한 줄에 한 문장

문장이 끝나는 ; 뒤에는 새줄을 삽입한다. 한 줄에 여러 문장을 쓰지 않는다.  
*- 나쁜 예*

|  |
| --- |
| id: number; name: string; |

*- 좋은 예*

|  |
| --- |
| id: number;  name: string; |

## let 보단 const 사용

const를 활용함으로써 해당변수가 재선언, 재할당이 안됨을 표현되므로 const를 우선적으로 사용한다.

# 들여쓰기 (Indentation)

들여쓰기는 코드의 계층을 구분하기 위해 추가하는 문자이다.

## 탭의 크기는 2개의 스페이스

1개의 탭의 크기는 스페이스 2개와 같도록 에디터에서 설정한다.

## 블럭 들여쓰기

메서드, 함수, 제어문 등의 코드 블럭이 생길 때마다 1단계를 더 들여쓴다.

# 중괄호 (Braces)

중괄호({,}) 는 클래스, 메서드, 제어문의 블럭을 구분한다.

## K&R 스타일로 중괄호 선언

클래스 선언, 메서드 선언, 조건/반복문 등의 코드 블럭을 감싸는 중괄호에 적용되는 규칙이다. 중괄호 선언은 K&R 스타일(Kernighan and Ritchie style)을 따른다. 줄의 마지막에서 시작 중괄호`{`를 쓰고 열고 새줄을 삽입한다. 블럭을 마친후에는 새줄 삽입 후 중괄호를 닫는다.  
*- 나쁜 예*

|  |
| --- |
| if (exp == null)  {  return false;  }  for (const key in obj)  {  ....  } |

*- 좋은 예*

|  |
| --- |
| if (exp == null) {  return false;  }  for (const key in obj) {  ....  } |

## 닫는 중괄호와 같은 줄에 else, catch, finally, while 선언

아래의 키워드는 닫는 중괄호(}) 와 같은 줄에 쓴다.  
else,  
catch,  
do-while 문에서의 while  
*- 나쁜 예*

|  |
| --- |
| if (line.startWith(WARNING\_PREFIX)) {  return LogPattern.WARN;  }  else if (line.startWith(DANGER\_PREFIX)) {  return LogPattern.DANGER;  }  else {  return LogPattern.NORMAL;  } |

*- 좋은 예*

|  |
| --- |
| if (line.startWith(WARNING\_PREFIX)) {  return LogPattern.WARN;  } else if (line.startWith(DANGER\_PREFIX)) {  return LogPattern.NORMAL;  } else {  return LogPattern.NORMAL;  } |

## 빈 블럭에 새줄 없이 중괄호 닫기 허용

내용이 없는 블럭을 선언할 때는 같은 줄에서 중괄호를 닫는 것을 허용한다.

*- 좋은 예*

|  |
| --- |
| interface ElementClass extends Vue {} |

## 조건/반복문에 중괄호 필수 사용

조건, 반복문이 한 줄로 끝더라도 중괄호를 활용한다. 이 문서에 언급된 중괄호의 전후의 공백, 제어문 앞 뒤의 새줄 규칙도 함께 고려한다.  
*- 나쁜 예*

|  |
| --- |
| if (exp == null) return false;  for (char ch : exp.toCharArray()) if (ch == 0) return false; |

*- 좋은 예*

|  |
| --- |
| if (exp == null) {  return false;  }  for (const key in obj) {  if (key == 0) {  return false;  }  } |

# 공백 (Whitespace)

## 중괄호의 시작 전, 종료 후에 공백 삽입

여는 중괄호({) 앞에는 공백을 삽입한다. 닫는 중괄호(}) 뒤에 else ,catch 등의 키워드가 있을 경우 중괄호와 키워드 사이에 공백을 삽입한다.  
*- 나쁜 예*

|  |
| --- |
| if(line.startsWith(WARN\_PREFIX)){  ...  }else{  ...  } |

*- 좋은 예*

|  |
| --- |
| if (line.startsWith(WARN\_PREFIX)) {  ...  } else {  ...  } |

## 제어문 키워드와 여는 소괄호 사이에 공백 삽입

if, for, while, catch, switch`와 같은 제어문 키워드의 뒤에 소괄호((,)`)를 선언하는 경우, 시작 소괄호 앞에 공백을 삽입한다.  
*- 나쁜 예*

|  |
| --- |
| if(maxLine > LIMITED){  return false;  } |

*- 좋은 예*

|  |
| --- |
| if (maxLine > LIMITED) {  return false;  } |

## 식별자와 여는 소괄호 사이에 공백 미삽입

식별자와 여는 소괄호(() 사이에는 공백을 삽입하지 않는다. 생성자와 메서드의 선언, 호출, 애너테이션 선언 뒤에 쓰이는 소괄호가 그에 해당한다.  
*- 나쁜 예*

|  |
| --- |
| const data = await getHellowText ();  this.setHelloText (data.data.helloText); |

*- 좋은 예*

|  |
| --- |
| const data = await getHellowText();  this.setHelloText(data.data.helloText); |

## 콤마/구분자 세미콜론의 뒤에만 공백 삽입

콤마(,)와 반복문(while, for)의 구분자로 쓰이는 세미콜론(;)에는 뒤에만 공백을 삽입한다.  
*- 나쁜 예*

|  |
| --- |
| for (let i = 0;i < stringLength;i++) {  display(level,message,i)  } |

*- 좋은 예*

|  |
| --- |
| for (let i = 0; i < length; i++) {  display(level, message, i)  } |

## 콜론의 앞 뒤에 공백 삽입

반복문과 삼항연산자에서 콜론(:)의 앞 뒤에는 공백을 삽입한다. 라벨 선언 뒤에는 아무런 문자열이 없으므로 앞에만 공백을 삽입한다.  
*-- 좋은 예*

|  |
| --- |
| process.env.NODE\_ENV === 'production' ? 'warn' : 'off' |

## 이항/삼항 연산자의 앞 뒤에 공백 삽입

이항/삼항 연산자의 앞 뒤에는 공백을 삽입한다.  
*- 좋은 예*

|  |
| --- |
| if (pattern == Access.ABNORMAL) {  return 0;  }  finalScore += weight \* rawScore - absentCount;  if (finalScore > MAX\_LIMIT) {  return MAX\_LIMIT;  } |

## 단항 연산자와 연산 대상 사이에 공백을 미삽입

단항 연산자와 연산 대상의 사이에는 공백을 삽입하지 않는다.  
전위 연산자 : 연산자 뒤에 공백을 삽입하지 않는다.  
전위 증감/감소 연산자 : ++,--  
부호로 쓰이는 +, -  
NOT 연산자 : ~, !  
후위 연산자 : 연산자 앞에 공백을 삽입하지 않는다.  
후위 증감/감소 연산자 : ++,--  
*- 나쁜 예*

|  |
| --- |
| point: number = ++ score \* rank-- \* -1; |

*- 좋은 예*

|  |
| --- |
| point: number = ++score \* rank-- \* -1; |

## 주석문 기호 전후의 공백 삽입

주석의 전후에는 아래와 같이 공백을 삽입한다.  
명령문과 같은 줄에 주석을 붙일 때 // 앞  
주석 시작 기호 // 뒤  
주석 시작 기호 /\* 뒤  
블록 주석을 한 줄로 작성시 종료 기호 \*/ 앞  
*- 좋은 예*

|  |
| --- |
| /\*  \* 공백 후 주석내용 시작  \*/  /\* 주석내용 앞에 공백, 뒤에도 공백 \*/ |

# 기타

## 표기법

*-* 카멜 케이스: 첫글자는 소문자로 시작하며 두번쨰 단어부터는 대문자로 표현하는 방식

|  |
| --- |
| myApplicationName |

*-* 파스칼 케이스: 첫글자는 대문자로 시작하며 단어와 단어 사이를 대문자를 이용하여 구분하는 방식

|  |
| --- |
| MyApplicationName |

*-* 케밥 케이스: 첫글자는 소문자로 시작하며 단어와 단어 사이에는 하이픈(-)으로 표현하는 방식

|  |
| --- |
| my-application-name |

*-* 스네이크 케이스: 첫글자는 소문자로 시작하며 단어와 단어 사이에는 언더바(\_)로 표현하는 방식

|  |
| --- |
| my\_application\_name |

# 참고자료

vuejs official style guide : <https://vuejs.org/style-guide>

google typescript style guide : <https://google.github.io/styleguide/tsguide.html>

vite github : <https://github.com/vitejs/vite>