Day4

자료	Vue
구분	Vue

components

컴포넌트는 Vue.js, React.js 등의 프론트엔드 프레임워크를 사용하는 가장 큰 이유라고 생각한다.

인스타그램 웹 페이지다.

프레임워크 없이 HTML, CSS, JavaScript 만으로 개발을 한다고 가정하자.

하나의 페이지는 하나의 HTML 만 존재한다. 만약 페이지 내용이 복잡해진다면 HTML 파일 하나만

몇천줄이 되는 것도 충분히 가능하다.

이것은, 유지 보수 관점에서 보면 최악이다. 두 가지 상황을 가정하겠다.

첫번째 상황.

인스타그램 팀에서, 오른쪽 위의 하트 아이콘을 별모양 아이콘으로 바꾼다고 가정하자.

1. `index.html` 파일을 연다.
2. 몇천줄의 코드에서 하트를 찾는다.
3. 다른 아이콘으로 교체한다.

문제 없어 보이지만, 하트 하나 교체하려고 몇천줄을 뒤적이는게, 과연 안전한 일일까?

두번째 상황.

당신은 인스타그램 팀에 입사한 신입이다. 디자이너는 당신에게 `<footer>` 태그에 들어갈 새 아이콘 리스트를 줬고, 다른 선배 개발자들은 메인 피드 영역을 작업 중이다.

1. `index.html` 을 가지고 한쪽에서는 `<footer>` 를 작업한다.
2. `index.html` 을 가지고 다른 한쪽에서는 피드를 작업하고, 서버에 업로드했다.
3. 그러나, `<footer>` 를 작업하는 쪽에서 이를 모르고 작업 종료 후 서버에 업로드했다.
4. 즉, `<footer>` 는 변경 적용 되었지만, 피드 변경은 적용되지 않았다!

이러한 문제를 방지하기 위해, 수정한 코드만 팀장에게 준다던가, 깃 사용 시 주의 한다던가 다양한 방법이 연구되었고, 실제로 그런 식으로 작업을 해왔다.

그러나, 어떤 똑똑한 사람이 이렇게 생각한 것이다.

하나의 페이지를 여러개의 파일로 나눠버리면 되지 않아?

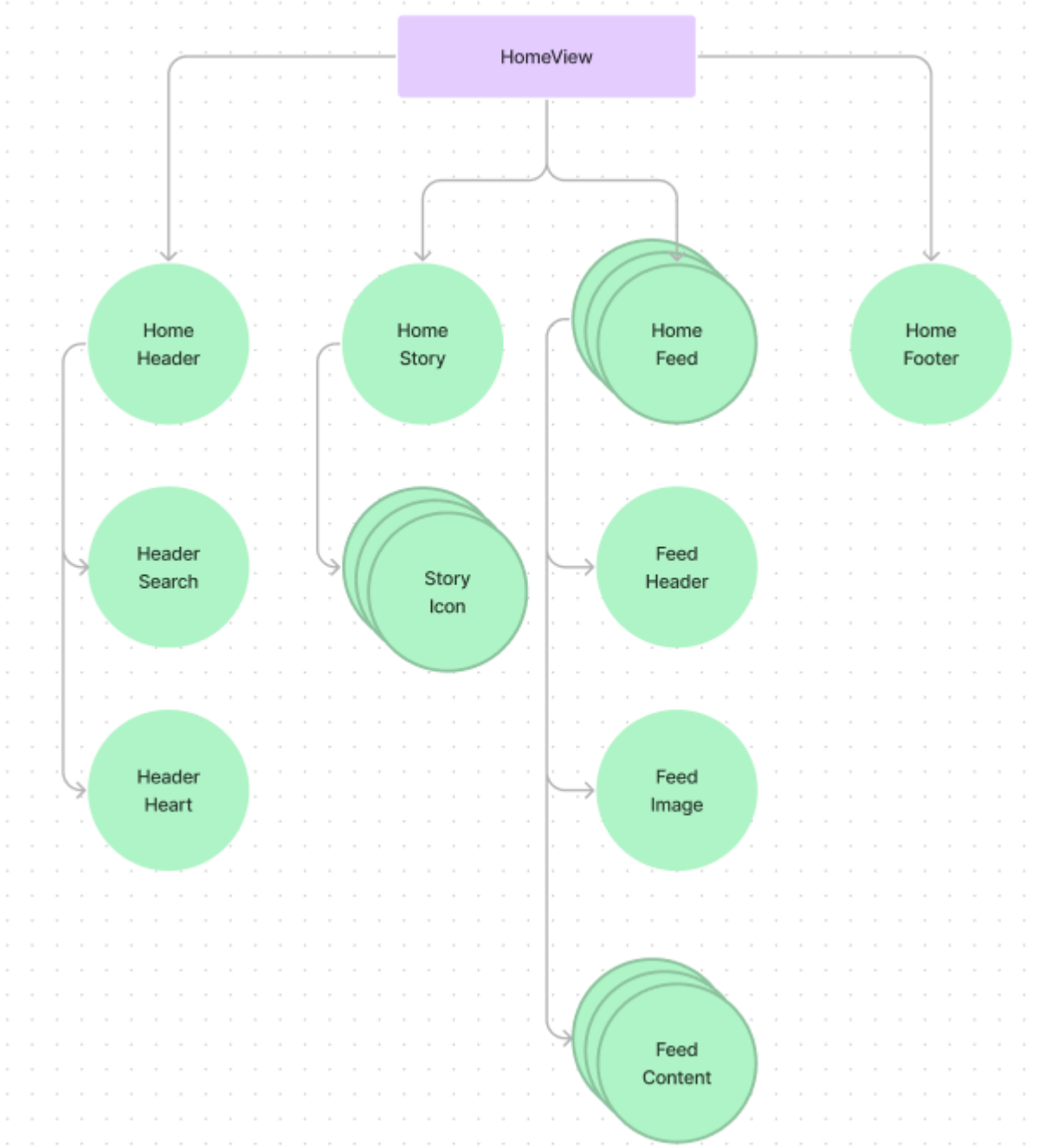
그게 바로 컴포넌트다. 레고를 생각하면 쉬운데, 각자 부품을 만들고, 싹 다 모아서 조립한다.

방금 전의 화면은 다음과 같이, 총 네 개의 컴포넌트로 나뉠 수 있다.

각각의 컴포넌트는 각각의 `.vue` 파일이 되는데, 각각 `AppHeader.vue` , `AppStory.vue` , `AppFeed.vue` , `Footer.vue` 로 이름 짓겠다.

그리고 각각의 컴포넌트는 필요에 따라서 각각의 자식 컴포넌트를 둘 수 있다.

즉, 다음과 같은 트리 형태를 띠게 된다.



각각의 트리 요소는 하나의 파일, 즉 하나의 컴포넌트다.

- 자세히 보면, `HomeFeed` 나 `StoryIcon` 처럼 여러개가 겹쳐져 있는 컴포넌트가 보이는데, 이는 `v-for` 를 통한 반복을 의미한다.
- 컴포넌트 이름은 루트 컴포넌트 `App` 을 제외하고 다음의 원칙에 따라 짓는다.
  - 두 개의 명사 결합
  - 대문자로 시작
  - 두 개의 명사를 생각하기 어렵다면, 첫번째 명사는 부모를 나타낼 수 있도록 할 것.

ex) `StoryIcon` ⇒ 부모가 `HomeStory`컴포넌트

이런 컴포넌트 방식을 따를 경우, 처음 언급했던 두 가지 문제가 해결된다.

첫번째 상황.

인스타그램 팀에서, 오른쪽 위의 하트 아이콘을 별모양 아이콘으로 바꾼다고 가정하자.

1. `HeaderHeart.vue` 파일을 연다.
2. 하트를 바꾼다.
3. 다른 컴포넌트는 건드리지 않는다.

두번째 상황.

당신은 인스타그램 팀에 입사한 신입이다. 디자이너는 당신에게 `<footer>` 태그에 들어갈 새 아이콘 리스트를 줬고, 다른 선배 개발자들은 메인 피드 영역을 작업 중이다.

1. 한쪽에서는 `HomeFooter.vue` 를 작업한다.
2. 다른 한쪽에서는 `HomeFeed.vue` 를 작업한다.
3. 별개의 파일을 작업하므로 안전하다. 각각 작업한 두 개의 컴포넌트 파일을 변경하고 업로드한다.

즉, 컴포넌트 방식으로 개발할 때, 유지보수와 협업이 유리해진다.

직접 자식 컴포넌트를 만들어보자.

먼저 `HomeView.vue` 를 다음과 같이 만든다.

```
<template>
  <h1>Home</h1>
</template>
```

- `HomeView.vue` 처럼, 뒤에 `.vue` 확장자가 붙어 있으면 하나의 컴포넌트로 취급한다. 컴포넌트를 지칭할때는 뒤에 확장자를 생략하고 `HomeView` 라고 부르는 경우가 흔하다.
- 특히 `HomeView` 처럼 `views/` 안에 위치하면서, `-View` 라고 네이밍된 컴포넌트는 라우트 컴포넌트라고 하며, 라우터 챕터에서 그 차이를 상세히 설명한다.

자식 컴포넌트 만들기 1단계: 파일을 만든다.

`HomeView` 컴포넌트는 총 세 개의 자식 컴포넌트를 가질 것이다.

`src/` 디렉터리에 `components/` 디렉터리 안에 `HomeHeader.vue` , `HomeMain.vue` , `HomeFooter.vue` 를 아래 양식으로 만든다.

```
<template>
  <h2>현재 컴포넌트 이름</h2>
</template>
```

자식 컴포넌트 만들기 2단계: 자식을 `import` 한다.

`HomeView`컴포넌트 맨 위에, `<script setup>` 태그를 생성한 후 다음과 같이 작성한다.

```
<script setup>
import HomeHeader from "@/components/HomeHeader.vue";
import HomeMain from "@/components/HomeMain.vue";
import HomeFooter from "@/components/HomeFooter.vue";
</script>

<template>
  <h1>Home</h1>
</template>
```

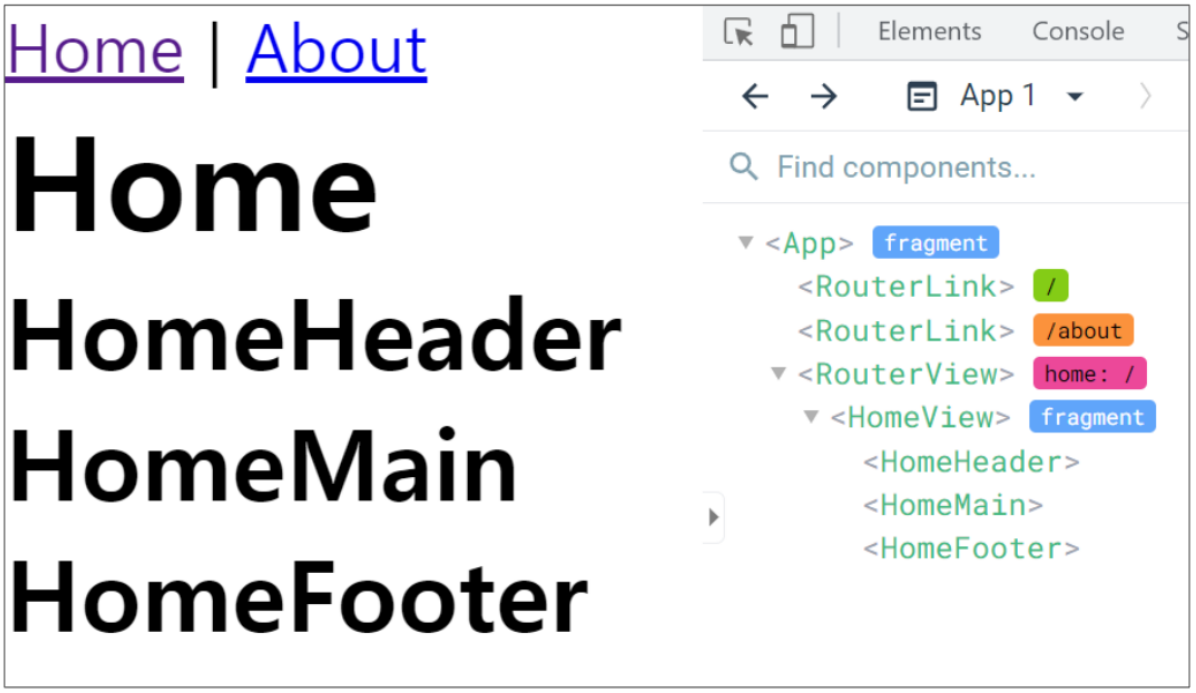
- `@` 는 `src/` 디렉터리를 뜻하는 약어다.

자식 컴포넌트 만들기 3단계: 자식 컴포넌트를 부모에 붙인다.

```
<script setup>
import HomeHeader from "@/components/HomeHeader.vue";
import HomeMain from "@/components/HomeMain.vue";
import HomeFooter from "@/components/HomeFooter.vue";
</script>

<template>
  <h1>Home</h1>
  <HomeHeader />
  <HomeMain />
  <HomeFooter />
</template>
```

컴포넌트를 붙일 땐, `<template>` 에서 일반 태그와 구분 되도록 PascalCase 를 사용하며, 생성과 함께 닫아준다.



Vue 개발자 도구를 열어서 확인해보면, 부모 자식 관계가 잘 설정된 것을 확인할 수 있다.

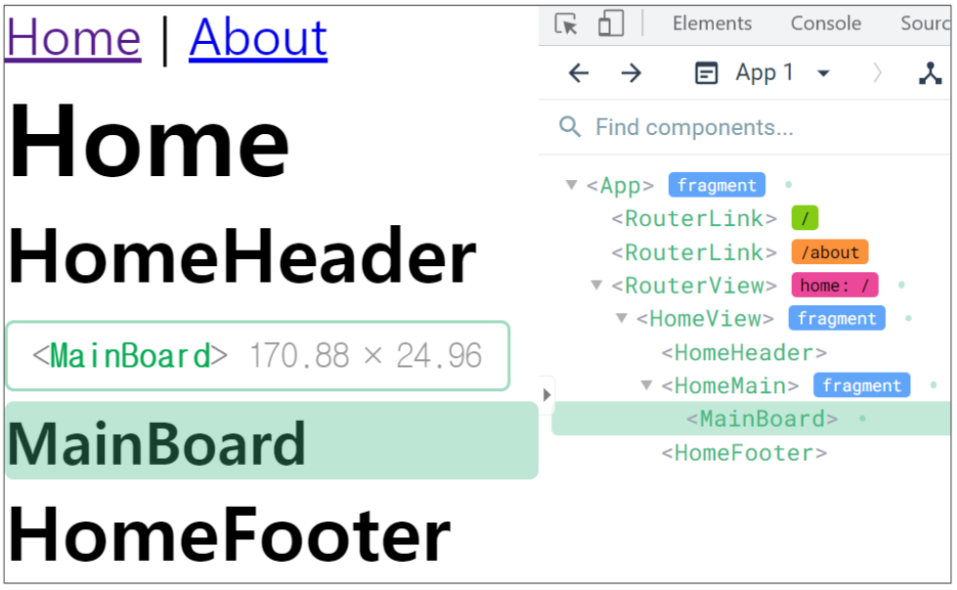
[도전과제]

도전: `HomeMain` 컴포넌트에, `MainBoard` 라는 자식 컴포넌트를 만들어 아래 그림과 같이 나오도록 해보자.

```
<template>
  <h3>MainBoard</h3>
</template>
```

(HomeView 컴포넌트에 자식을 만드는 것이 아닙니다!! HomeMain 입니다.)

(h3 태그로 만들었습니다)



scoped

scoped 에 대해서 알아보자.

하나의 컴포넌트는 기본적으로 다음과 같은 구조를 지닌다.

```
<script setup>
// js code
</script>

<template>
<!-- html code -->
</template>

<style scoped>
```

```
/* css code */
</style>
```

이 기본적인 구조는 반드시 알아둬야 한다.

`<template>` : HTML

`<script setup>` : JavaScript

`<style scoped>` : CSS

즉, 하나의 컴포넌트에서 별도의 파일 분리를 하지 않고, HTML, CSS, JavaScript 를 같이 사용한다.

그런데, `<style>` 태그 안에 `scoped` 속성은 대체 무슨 뜻일까?

`scoped` 가 없다면 전역 스타일링.

`scoped` 가 있다면 지역 스타일링 이며 일반적으로 컴포넌트에서 붙여서 사용한다.

차이를 직접 확인 해 보자.

먼저, `HomeView` 컴포넌트에서 `<style>` 태그를 `scoped` 없이 작성하겠다.

아래 코드를 참고하자.

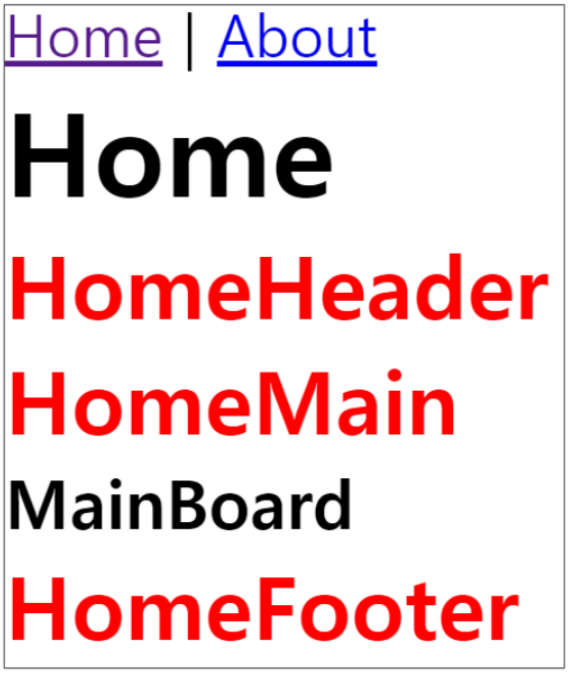
```
<script setup>
import HomeHeader from "@/components/HomeHeader.vue";
import HomeMain from "@/components/HomeMain.vue";
import HomeFooter from "@/components/HomeFooter.vue";
</script>

<template>
  <h1>Home</h1>
  <HomeHeader />
  <HomeMain />
  <HomeFooter />
</template>

<style>
h2 {
  color: red;
}
</style>
```

무슨 뜻인가? 모든 `<h2>` 태그의 글자 색은 빨간색으로 하라는 뜻이다.

그런데 자세히 보면, `HomeView` 컴포넌트엔 `<h2>` 태그가 존재하지 않는다! 그러나, 결과는 다음과 같다.



자세히 보면, `<h2>` 태그를 가진 자식 컴포넌트의 모든 색깔은 빨간색으로 처리 되었다.

왜 이런 현상이 생기는 것일까? 수천 줄 수만 줄의 코드라도, Vue.js 프로젝트는 결국 하나의 HTML 문서로 (`index.html`)로 이루어져 있다. 이를 SPA, Single Page Application 이라고 하는데, 페이지가 단 하나라는 뜻이다.

이는 어떤 컴포넌트 파일이든 상관없이, 모든 컴포넌트 파일의 태그에 접근해서 스타일링할 수 있다는 것을 뜻한다. 현재 `<h2>` 태그는 각각 `HomeHeader` , `HomeMain`,`HomeFooter` 컴포넌트에 있으므로, 해당 태그는 모두 빨간색이 된다.

그렇다면, `scoped` attribute를(속성) 추가할 시 어떻게 될까? 이번엔 `HomeMain` 에서 `<style>` 태그를 만들되, `scoped` 를 붙여서 다음과 같이 만들어주자.

```
<script setup>
import MainBoard from "@/components/MainBoard.vue";
</script>

<template>
  <h2>HomeMain</h2>
  <MainBoard />
</template>

<style scoped>
h2 {
  color: blue;
}
</style>
```

`scoped` 를 붙이면, 스타일의 영역을 해당 컴포넌트로 한정 짓는다. 다른 컴포넌트에 `<h2>` 가 있든 말든, 오로지 현재 컴포넌트의 `<h2>` 에만 스타일링을 적용한다.

그럼 둘은 언제 쓰는 게 좋을까? 일반적인 사용법은 다음과 같다.

`<style>` : 루트 컴포넌트(`App.vue`)에서 전역 스타일링 할 때 사용

`<style scoped>` : 일반 컴포넌트에서 사용

ex) 만약 모든 폰트를 Noto Sans KR 로 바꾸고 싶다면 `App.vue` 에서 `scoped` 없이 적용 하는것이 적절할 것이다.

- Vite 에서 기본적으로 만들어주는 템플릿을 자세히 보면, `App.vue` 에서도 `scoped` 를 붙여 지역 스타일링을 하는 것을 볼 수 있으며, 전역 스타일링을 할 일이 있다면 `src/assets/main.css` 에서 작업하는 방식을 취하는 것을 알 수 있다.
- 따라서 앞으로 이 방식을 따라, 모든 컴포넌트에 `scoped` 를 붙여 지역 스타일링하고, 전역 스타일링할 일이 있다면 `main.css` 에서 작업하도록 하겠다.
- 다만 지금은 `scoped` 가 무슨 뜻인지 설명하기 위해서 존재한다. <끝>