

Component

Vue-CLI

Component

Single File Component

- Component가 전역으로 선언되었을 때 불편한 점들
 - 각 Component들마다 이름을 다르게 지어야 한다.
 - template 안의 코드를 복잡하게 디자인하기 어렵다.
 - html과 js는 모듈화 되어있지만 css는 아니다.
- Template마다 파일을 분리해 모듈화 한다면 편리하게 사용할 수 있다.

설치

- node.js 8.9 이상 필요
- 이전 버전의 vue-cli가 이미 설치되어 있다면 `npm uninstall vue-cli -g` 실행
- `npm install -g @vue/cli`

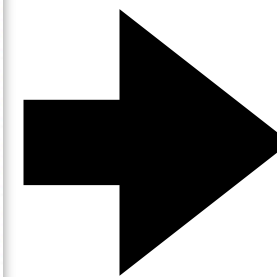
프로젝트 만들기

- 다음 명령어로 새 프로젝트를 생성

`vue create hello-vue`

- hello-vue 부분이 앱 이름, 이후 화면에서 Defalut 에서 Enter, 프로젝트 폴더에서 실행 가능

```
Vue CLI v4.5.13
? 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 hello-vue.
👉 Get started with the following commands:
```

```
$ cd hello-vue
$ npm run serve
```

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

실행

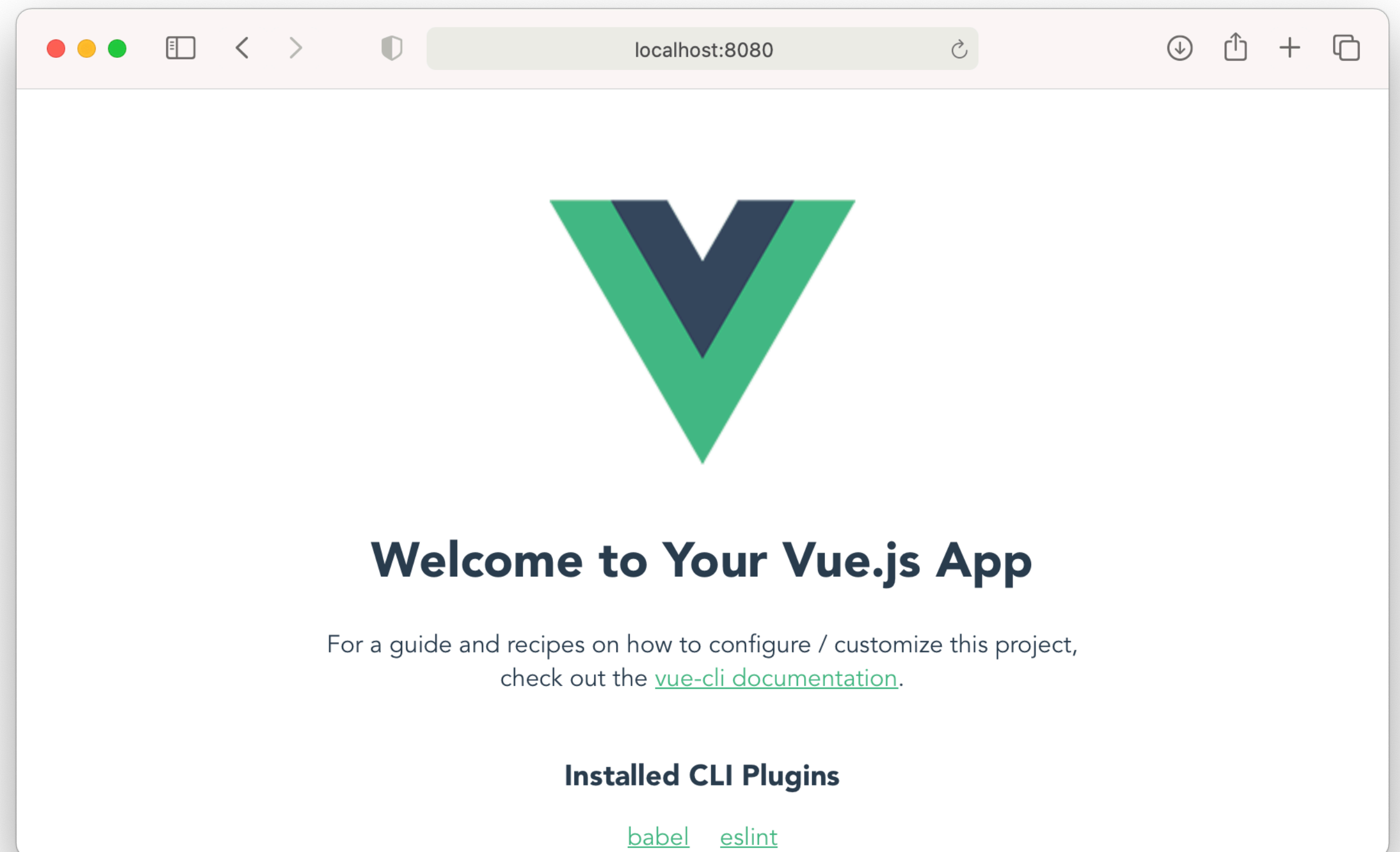
- `cd hello-vue`
- `npm run serve`

DONE Compiled successfully in 959ms

App running at:

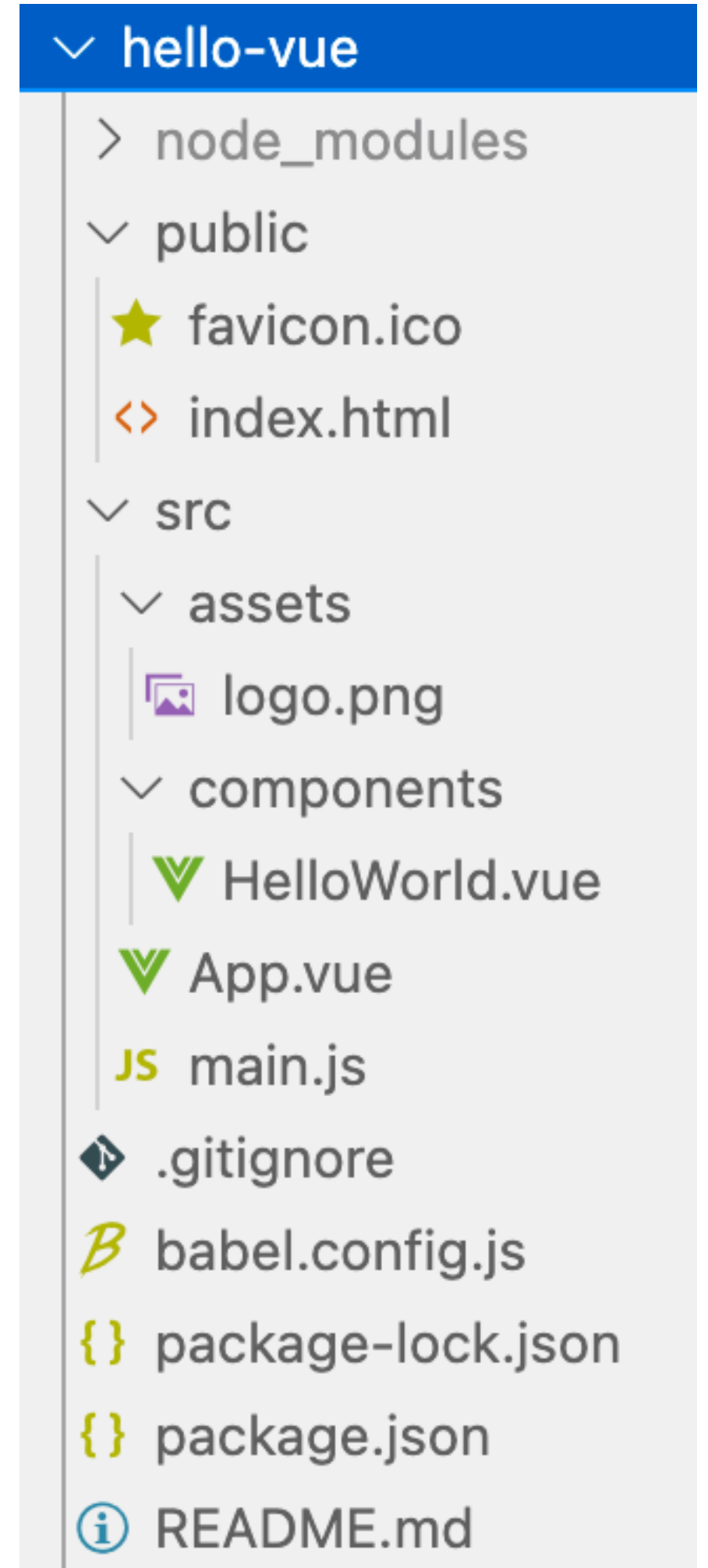
- Local: <http://localhost:8080/>
- Network: <http://192.168.219.105:8080/>

Note that the development build is not optimized.
To create a production build, run `npm run build`.



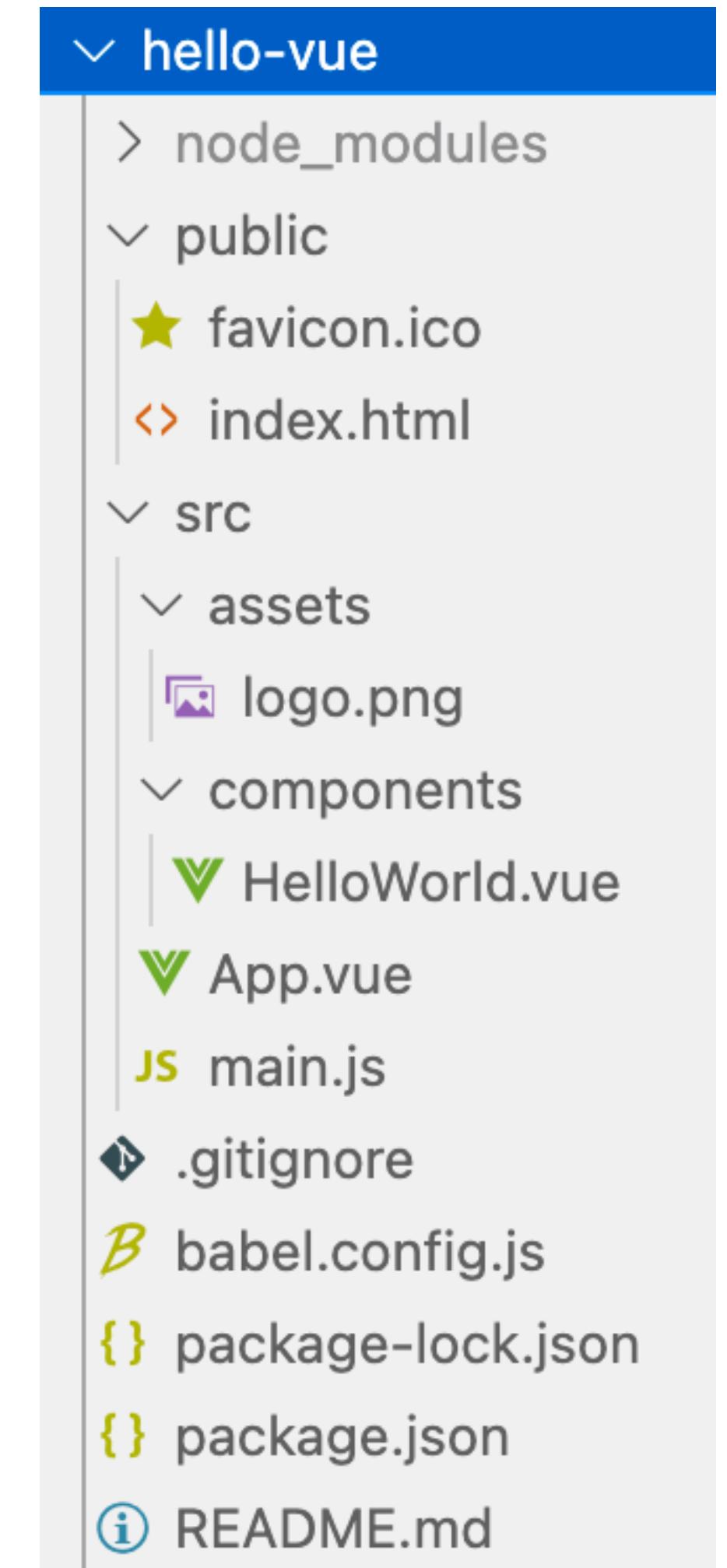
구조 살펴보기

- node_modules: 라이브러리 설치 폴더
- public: webpack이 컴파일 하지 않는 파일들.
- src: vue 작업 폴더



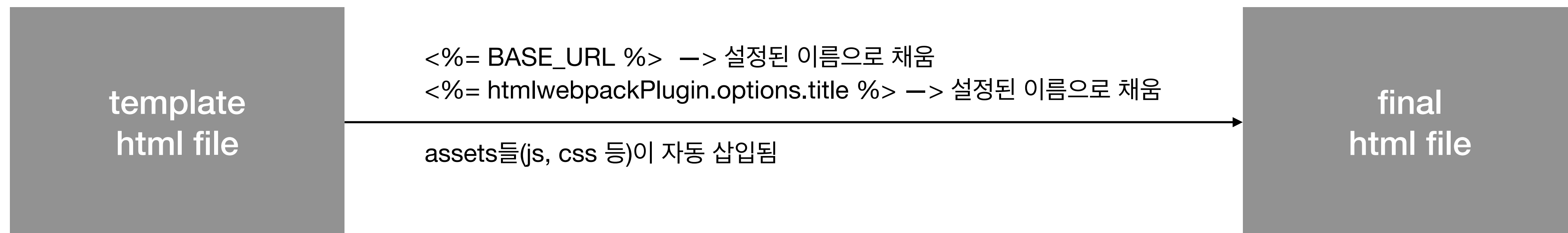
구조 살펴보기

- 프로젝트 루트 폴더의 vue.config.js에 정의된 내용을 기준으로 실행
- 해당 파일이 정의되지 않은 경우 기본 설정에 따라 실행됨.
- webpack.config.js는 vue-cli-service 로 숨겨놓음
 - 현재 옵션 보기: vue inspect > options.js
 - 옵션 수정은 vue.config.js 수정
- <https://cli.vuejs.org/config/#vue-config-js>



config 파일 살펴보기

- entry: 앱의 시작점
- template: 시작 페이지
 - HtmlWebpackPlugin 을 이용해 설정된 html을 이용해 동적인 html을 만들어줌



config 파일 수정하기

- 프로젝트 루트에 vue.config.js 파일을 생성 후 수정 가능

- <https://cli.vuejs.org/config/#pages>

```
module.exports = {  
  pages: {  
    index: {  
      // entry for the page  
      entry: 'src/index/main.js',  
      // the source template  
      template: 'public/index.html',  
      // output as dist/index.html  
      filename: 'index.html',  
      // when using title option,  
      // template title tag needs to be <title><%= htmlWebpackPlugin.options.title  
      title: 'Index Page',  
      // chunks to include on this page, by default includes  
      // extracted common chunks and vendor chunks.  
      chunks: ['chunk-vendors', 'chunk-common', 'index']  
    },  
    // when using the entry-only string format,  
    // template is inferred to be `public/subpage.html`  
    // and falls back to `public/index.html` if not found.  
    // Output filename is inferred to be `subpage.html`.  
    subpage: 'src/subpage/main.js'  
  }  
}
```

public 폴더

- public 폴더의 리소스들은 webpack을 통과하지 않고 전달됨.
 - 절대 경로로 사용해야 함.
 - webpack의 번들에 포함되지 않고 최적화 되지 않음.
 - html에서 사용할 때는 `<%= BASE_URL %>`파일이름 으로
 - vue의 data에서는 `process.env.BASE_URL`로 접근

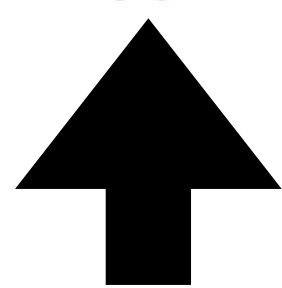
다시 보기

- main.js
 - template으로 설정된 index.html에 어떤 element를 붙여야 하는지를 설정.
- App.vue 최상위 컴포넌트
- components: 컴포넌트 작업 폴더
- assets: 이미지, css. src 폴더 하단에 있는 이 파일들은 webpack이 처리한다.

```
import Vue from 'vue'
import App from './App.vue'

Vue.config.productionTip = true

new Vue({
  |   render: h => h(App),
  }).$mount('#app')
```



```
new Vue({
  |   render: function (createElement) {
  |     |   return createElement(App)
  |   }
  }).$mount('#app');
```

App.vue

Single File Component

- template, script, style으로 나뉘어 하나의 모듈을 정의
- template: template 언어를 정의할 수 있음, html에 해당하는 부분
- script: template과 함께 사용될 스크립트 정의
 - import 를 이용해 sub component를 내 template에서 사용할 수 있음
- style: css를 정의하는데 사용. scoped를 정의할 경우 이 컴포넌트에만 적용됨

실습 예제를 다시 만들기

양말 예제 다시 만들기

- vue create my-socks
- src/assets 폴더에 images 폴더 만들고 이미지 넣기
 - socks_green.jpg, socks_blue.jpg
- src/assets 폴더에 css 폴더 만들고 styles.css 파일 넣기

양말 예제 다시 만들기

- src/components 폴더에 Product.vue 파일 추가
 - 상품 정보 페이지
- src/components 폴더에 ProductPage.vue 파일 추가
 - 상품 정보 페이지와 카트를 포함하는 페이지.

Product.vue

지난 작업 내용을 이용해 작성

```
<template>
  <div class="product">
    <div class="product-image">
      
    </div>
    <div class="product-info">
      <h1>{{ product }}</h1>
      <p v-if="inStock">In Stock</p>
      <p v-else>Out of Stock</p>
      <p>Shipping: {{ shipping }}</p>
      <ul>
        <li v-for="(detail, i) in details" :key="i">{{ detail }}</li>
      </ul>
      <div class="color-box"
        v-for="(variant, index) in variants"
        :key="variant.variantId"
        :style="{ backgroundColor: variant.variantColor }"
        @mouseover="updateProduct(index)"
      >
      </div>
      <button v-on:click="addToCart"
        :disabled="!inStock"
        :class="{ disabledButton: !inStock }">
        Add to cart
      </button>
    </div>
  </div>
</template>
```

```
<script>
export default{
  data() {
    return {
      product: 'Socks',
      brand: 'Vue Mastery',
      selectedVariant: 0,
      details: ['80% cotton', '20% polyester', 'Gender-neutral'],
      variants: [
        {
          variantId: 2234,
          variantColor: 'green',
          variantImage: require('@/assets/images/socks_green.jpg'),
          variantQuantity: 10
        },
        {
          variantId: 2235,
          variantColor: 'blue',
          variantImage: require('@/assets/images/socks_blue.jpg'),
          variantQuantity: 10
        }
      ]
    }
  },
  methods: {
    addToCart: function() {
      this.$emit('add-to-cart', this.variants[this.selectedVariant].variantId)
    },
    updateProduct: function(index) {
      this.selectedVariant = index
    }
  },
  computed: {
    title() {
      return this.brand + ' ' + this.product
    },
    image(){
      return this.variants[this.selectedVariant].variantImage
    },
    inStock(){
      return this.variants[this.selectedVariant].variantQuantity
    },
    shipping() {
      if (this.premium) {
        return "Free"
      }

      return 2.99
    }
  }
};
</script>
```

ProductPage.vue

지난 작업 내용을 이용해 작성

```
<template>
  <div id="app">
    <div class="nav-bar"></div>
    <div class="cart">
      <p>Cart({{ cart.length }})</p>
    </div>
    <Product @add-to-cart="addToCart"></Product>
  </div>
</template>

<script>
import Product from './Product.vue';

export default{
  name: 'ProductPage',
  components:{
    Product
  },
  data(){
    return {cart: []}
  },
  methods:{
    addToCart(variantId){
      this.cart.push(variantId)
    }
  }
}
</script>
```

App.vue

```
<template>
  <div id="app">
    <ProductPage></ProductPage>
  </div>
</template>

<script>
import ProductPage from './components/ProductPage.vue'

export default {
  name: 'App',
  components: {
    ProductPage
  }
}
</script>

<style>
@import './assets/css/styles.css';
</style>
```

Simple todo

<https://codesandbox.io/s/o29j95wx9>

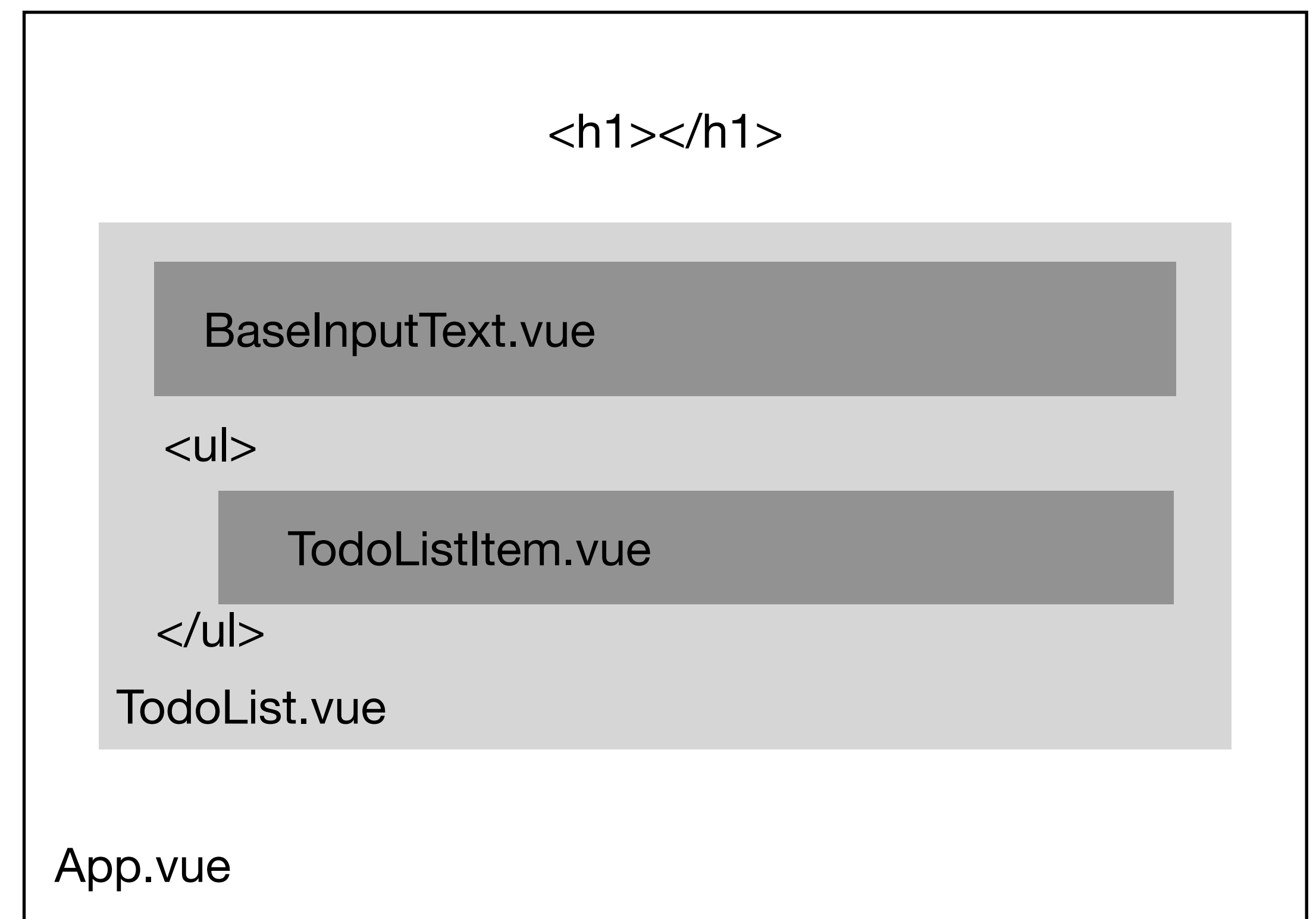
실습 목표

- hello-vue 프로젝트를 이용해 간단한 todo list 앱을 작성해본다.

My Todo App!

New todo

- Learn Vue ☒
- Learn about single-file components ☒
- Fall in love ☒



App.vue

components/BaseInputText.vue

- <input> 만을 가진다.

```
<template>
  <input type="text" class="input" :value="value" v-on="listeners">
</template>

<script>
export default {
  props: {
    value: { type: String, default: '' },
  },
  computed: {
    listeners () {
      return {
        // 자식에게서 부모에게 전달되는 모든 이벤트들
        ...this.$listeners,
        // input 이벤트만 재정의
        input: event => this.$emit('input', event.target.value)
      }
    }
  }
}
</script>

<style scoped>
.input {
  width: 100%;
  padding: 8px 10px;
  border: 1px solid #32485F;
}
</style>
```

components/TodoListItem.vue

- 리스트 각 항목을 출력할 템플릿
- 버튼을 클릭할 경우 'remove' 이벤트를 발생시킴

```
<template>
  <li>
    {{ todo.text }}
    <button @click="$emit('remove', todo.id)">X</button>
  </li>
</template>

<script>
export default {
  props: {
    todo: {
      type: Object,
      required: true
    }
  }
}
</script>
```


components/TodoList.vue

- template

```
<template>
  <div>
    <BaseInputText v-model="newTodoText" placeholder="New todo" @keydown.enter="addTodo"/>
    <ul v-if="todos.length">
      <TodoListItem v-for="todo in todos" :key="todo.id" :todo="todo" @remove="removeTodo"/>
    </ul>
    <p v-else>
      Nothing left in the list. Add a new todo in the input above.
    </p>
  </div>
</template>
```

components/ToDoList.vue

- script
 - BaseInputText에서 입력 받은 내용을 todos에 추가
 - remove event 발생 시 항목 삭제

```
<script>
import BaseInputText from './BaseInputText.vue'
import ToDoListItem from './ToDoListItem.vue'

let nextTodoId = 1

export default {
  components: {
    BaseInputText, ToDoListItem
  },
  data () {
    return {
      newTodoText: '',
      todos: []
    }
  },
  methods: {
    addTodo () {
      const trimmedText = this.newTodoText.trim()
      if (trimmedText) {
        this.todos.push({
          id: nextTodoId++,
          text: trimmedText
        })
        this.newTodoText = ''
      }
    },
    removeTodo (idToRemove) {
      this.todos = this.todos.filter(todo => {
        return todo.id !== idToRemove
      })
    }
  }
}
</script>
```

App.vue

- 제목과 전체 스타일 정의

```
<template>
  <div id="app">
    <h1>My Todo App!</h1>
    <TodoList></TodoList>
  </div>
</template>

<script>
import TodoList from './components/TodoList.vue'

export default {
  name: 'App',
  components: {
    TodoList
  }
}
</script>

<style>
*, *::before, *::after {
  box-sizing: border-box;
}

#app {
  max-width: 400px;
  margin: 0 auto;
  line-height: 1.4;
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  color: #32485F;
}

h1 {
  text-align: center;
}
</style>
```