Vue.js Intermediate

Reactivity System, Vuex, Proven Patterns, Real World Examples

Profile

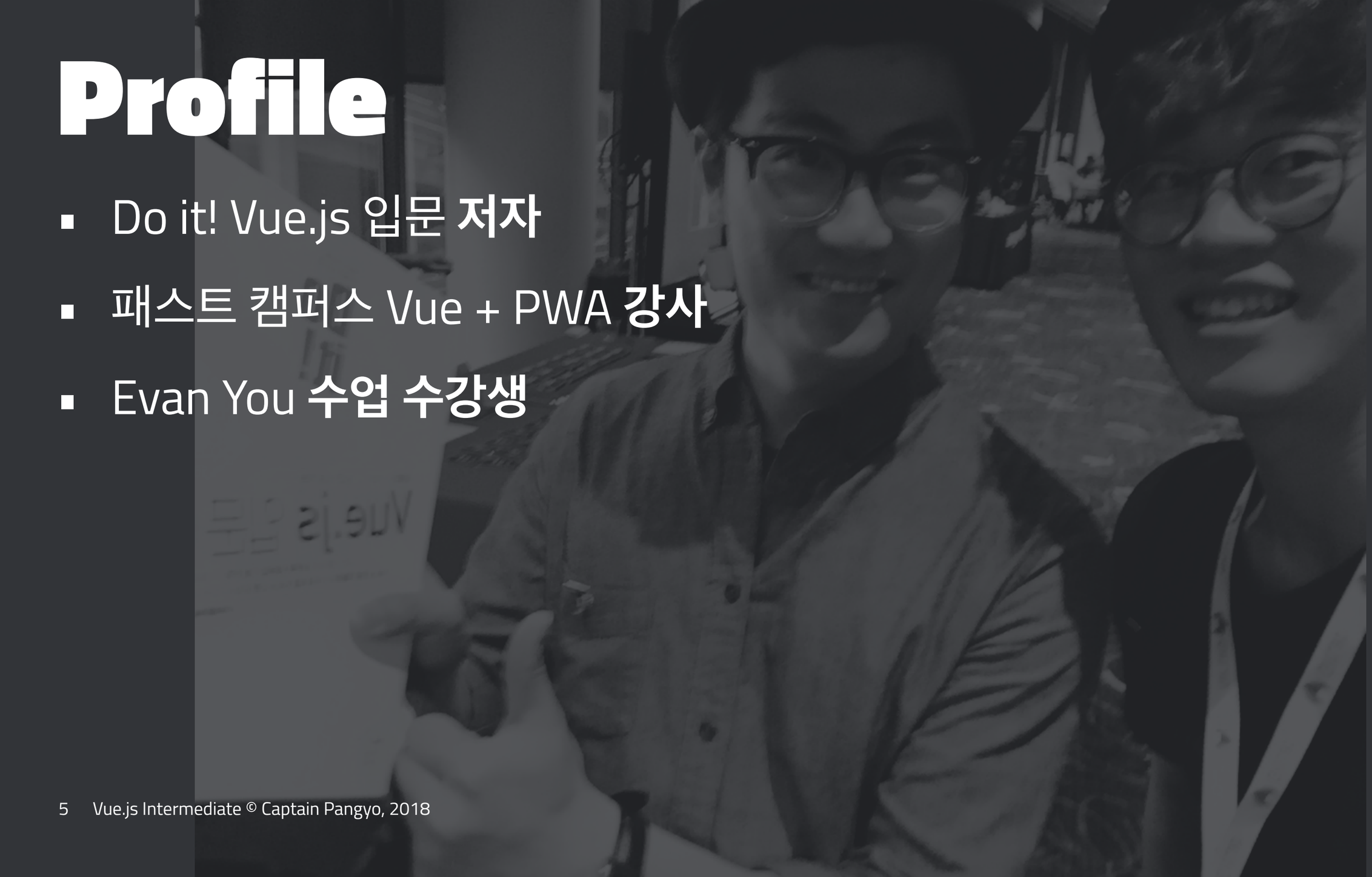
Profile

- Do it! Vue.js 입문 **저자**

Profile

- Do it! Vue.js 입문 **저자**
- 패스트 캠퍼스 Vue + PWA **강사**

Profile



- Do it! Vue.js 입문 **저자**
- 패스트 캠퍼스 Vue + PWA **강사**
- Evan You **수업 수강생**

Contents

- Vue Reactivity System 해부
- Vuex 기술요소와 모듈화
- Vue Proven Patterns
- Real World Examples

1. Vue Reactivity System

Reactivity

Reactivity

- JS의 데이터가 변경되면 화면의 DOM 요소가 자동으로 갱신되는 특징

Reactivity

- JS의 데이터가 변경되면 화면의 DOM 요소가 자동으로 갱신되는 특징
- 리액트의 Immutability와 대조되는 가장 큰 차이점

Reactivity

- JS의 데이터가 변경되면 화면의 DOM 요소가 자동으로 갱신되는 특징
- 리액트의 Immutability와 대조되는 가장 큰 차이점
- Vue 2.5 버전까지는 **`Object.defineProperty()`**로 반응성을 구현

Reactivity

- JS의 데이터가 변경되면 화면의 DOM 요소가 자동으로 갱신되는 특징
- 리액트의 Immutability와 대조되는 가장 큰 차이점
- Vue 2.5 버전까지는 **`Object.defineProperty()`**로 반응성을 구현
- Vue 3.x 버전부터는 ES6의 **`Proxy()`**로 반응성을 구현

ES5 - Object.defineProperty()

ES5 - Object.defineProperty()

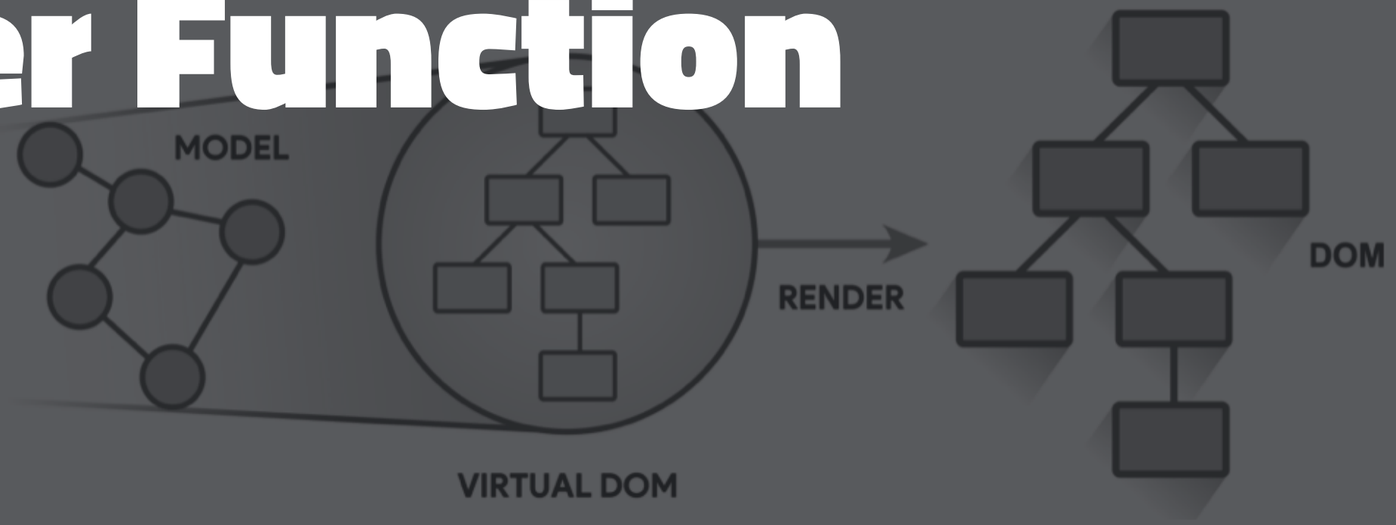
```
var sp1 = document.querySelector('.sp1');
var sp2 = document.querySelector('.sp2');
var viewModel = {};
var value = '10';

Object.defineProperty(viewModel, 'model', {
    get: function() {
        sp1.innerHTML = value;
    },
    set: function(newValue) {
        if (value !== newValue) {
            value = newValue;
            sp2.innerHTML = newValue;
        }
    }
});

/* get - 데이터 접근 */
viewModel.model;
/* set - 데이터 조작 */
viewModel.model = 30;
```

ES6 - Proxy()

ES6 - Proxy()

```
var sp1 = document.querySelector('.sp1');
var sp2 = document.querySelector('.sp2');
var viewModel = {};
var handler = {
  get: function(obj, prop) {
    console.log(obj[prop]);
    sp1.innerHTML = obj[prop];
  },
  set: function(obj, prop, value) {
    obj[prop] = value;
    sp2.innerHTML = value;
  }
};

var p = new Proxy(viewModel, handler);
/* 데이터 설정 */
p.model = 10;
/* 데이터 조회 */
p.model;
```

Render Function




```
<template>
  <div>
    <section>
      Do it! Vue.js 입문
    </section>
    <p>
      출간 기념 공개 세미나
    </p>
  </div>
</template>
```

```
render: function (createElement)
{
  return createElement('div', [
    createElement(
      'section',
      'Do it! Vue.js 입문'),
    createElement(
      'p',
      '출간 기념 공개 세미나')
  ]);
}
```

Render Function

- 뷰 템플릿을 화면에 뿌릴 때 프레임워크 내부적으로 변환 작업을 수행


```
<template>
  <div>
    <section>
      Do it! Vue.js 입문
    </section>
    <p>
      출간 기념 공개 세미나
    </p>
  </div>
</template>
```

```
render: function (createElement)
{
  return createElement('div', [
    createElement(
      'section',
      'Do it! Vue.js 입문'),
    createElement(
      'p',
      '출간 기념 공개 세미나')
  ]);
}
```

Render Function

- 뷰 템플릿을 화면에 뿌릴 때 프레임워크 내부적으로 변환 작업을 수행
- **`createElement('태그 이름', 내용)`** 로 구성


```
<template>
  <div>
    <section>
      Do it! Vue.js 입문
    </section>
    <p>
      출간 기념 공개 세미나
    </p>
  </div>
</template>
```

```
render: function (createElement)
{
  return createElement('div', [
    createElement(
      'section',
      'Do it! Vue.js 입문'),
    createElement(
      'p',
      '출간 기념 공개 세미나')
  ]);
}
```

Initial Render

Initial Render

1. Render Function 으로 변환

Initial Render

1. Render Function 으로 변환
2. Virtual DOM 반환

Initial Render

1. Render Function 으로 변환
2. Virtual DOM 반환
3. 실제 DOM 반환

Rendering Updates

Rendering Updates

1. 새로운 Virtual DOM 반환

Rendering Updates

1. 새로운 Virtual DOM 반환
2. 이전 Virtual DOM과 비교 후 업데이트

Rendering Updates

1. 새로운 Virtual DOM 반환
2. 이전 Virtual DOM과 비교 후 업데이트
3. 실제 DOM에 적용

Virtual DOM?

Virtual DOM?

- 고비용의 DOM 변경을 최소화하기 위한 기술

Virtual DOM?

- 고비용의 DOM 변경을 최소화하기 위한 기술
- DOM을 변경할 때마다 Reflow가 발생함 (뒤에 브라우저 동작과정 설명)

Actual DOM vs Virtual DOM

Actual DOM vs Virtual DOM

Actual DOM

```
document.createElement('div')
```

Actual DOM vs Virtual DOM

Actual DOM

```
document.createElement(‘div’)
```

Virtual DOM

```
vm.$createElement(‘div’)
```

Actual DOM vs Virtual DOM

Actual DOM

```
[object HTMLDivElement]
```

Actual DOM vs Virtual DOM

Actual DOM

```
[object HTMLDivElement]
```

Virtual DOM

```
{ tag: 'div', data: { attrs: {}, … }, children: []}
```

Actual DOM vs Virtual DOM

Actual DOM

```
[object HTMLDivElement]
```

브라우저 네이티브 객체 (고비용)

Virtual DOM

```
{ tag: 'div', data: { attrs: {}, … }, children: []}
```

Actual DOM vs Virtual DOM

Actual DOM

`[object HTMLDivElement]`
브라우저 네이티브 객체 (고비용)

Virtual DOM

`{ tag: 'div', data: { attrs: {}, … }, children: []}`
플레인 자바스크립트 객체 (저비용)

Virtual DOM

Virtual DOM

- 특정 시점의 실제 DOM의 모습을 담고 있는 자바스크립트 데이터 형식

Virtual DOM

- 특정 시점의 실제 DOM의 모습을 담고 있는 자바스크립트 데이터 형식
- 실제 DOM이 수행하던 렌더링 과정을 분리하여 다룰 수 있음

Virtual DOM

- 특정 시점의 실제 DOM의 모습을 담고 있는 자바스크립트 데이터 형식
- 실제 DOM이 수행하던 렌더링 과정을 분리하여 다룰 수 있음
- non 브라우저 환경에서도 렌더링을 할 수 있게 됨 (서버 사이드, 모바일)

Render Function

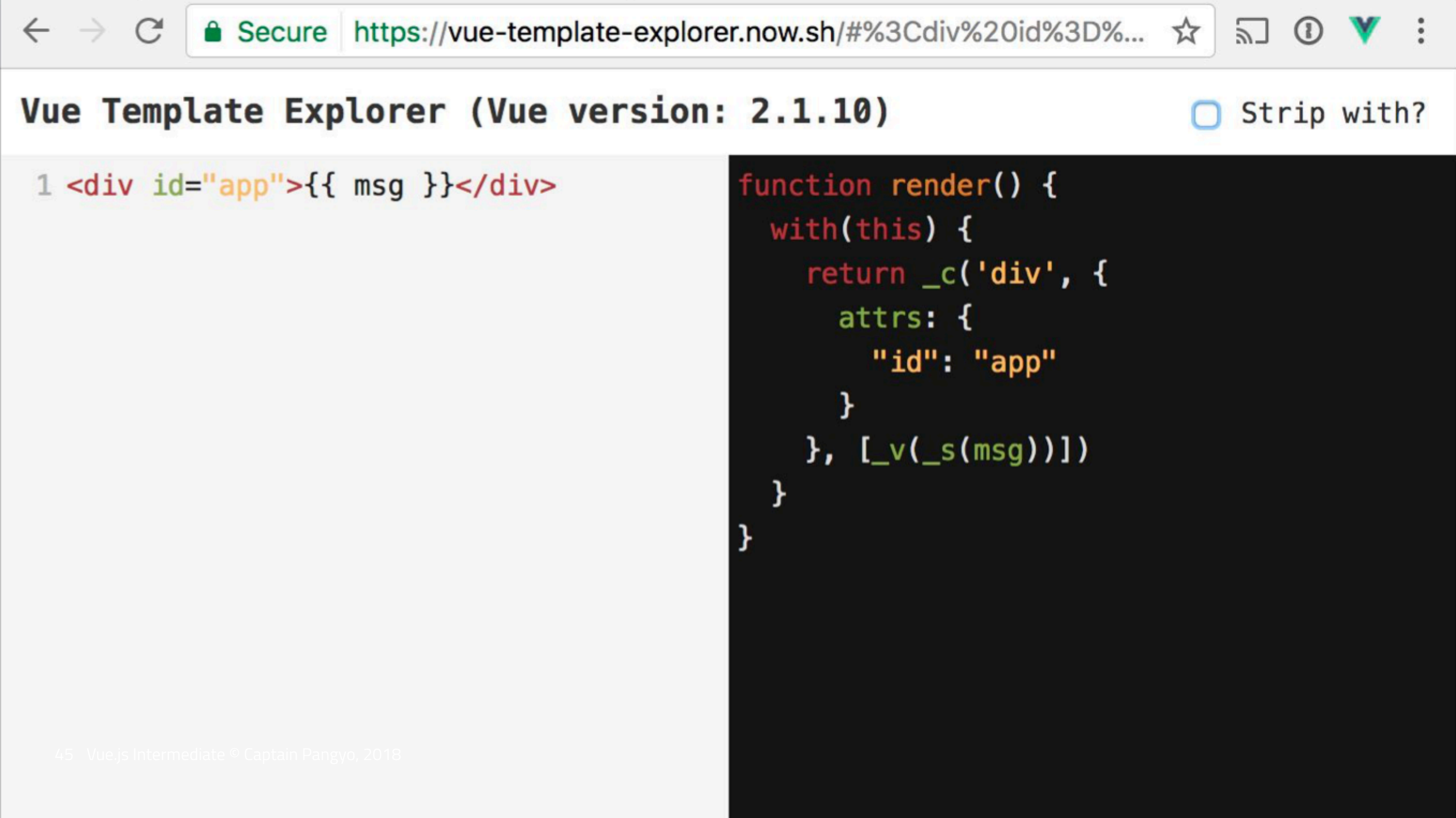
Render Function

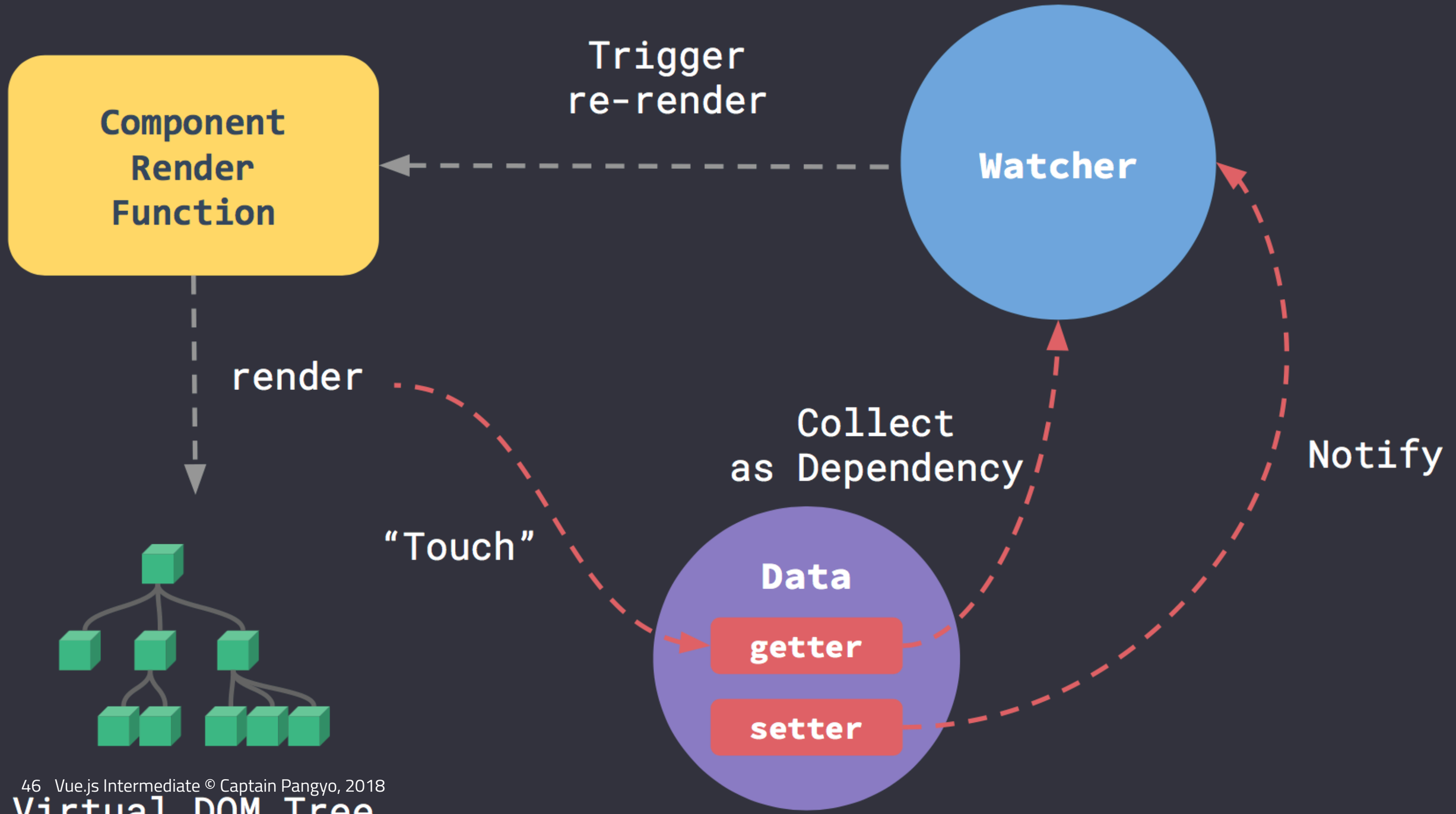
- Virtual DOM을 반환(생산)해주는 함수

Render Function

- Virtual DOM을 반환(생산)해주는 함수
- 템플릿 변환 과정에 관여하며 아래와 같은 절차를 따름

`Vue Template -> Compiler -> Render Function`


Secure
https://vue-template-explorer.now.sh/#%3Cdiv%20id%3D%...
Vue Template Explorer (Vue version: 2.1.10)
Strip with?
1 {{ msg }}
function render() {
with(this) {
return _c('div', {
attrs: {
"id": "app"
}
}, [_v(_s(msg))])
}
}


Component
Render
Function
Trigger
re-render
Watcher
render
Collect
as Dependency
Notify
"Touch"
Data
getter
setter
Virtual DOM Tree

Render Function API

```
export default {
    render(h) {
        return h('div', {}, […])
    }
}
```

The “h” function

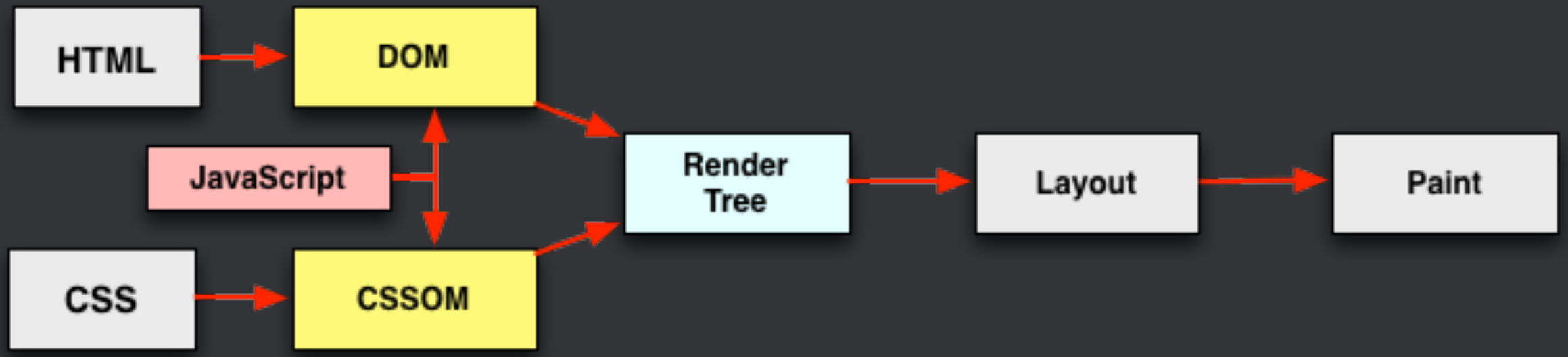
```
h(‘div’, ‘some text’)

h(‘div’, { class: ‘foo’ }, ‘some text’)

h(‘div’, {…}, [
    ‘some text’,
    h(‘span’, ‘another text’)
])
```

‘h’ can render a component

```
import MyComponent from ‘…’

h(MyComponent, {
    props: {…}
})
```

Browser Rendering Path

- Virtual DOM을 이해하기 위한 브라우저 렌더링 과정 분석



실습 1 - Render 함수 구현하기

실습 2 - Render 함수로 리스트 출력하기

2. State Management

Vuex?

Vuex?

- 복잡한 애플리케이션을 구현하기 위한 **상태 관리 패턴, 라이브러리**

Vuex?

- 복잡한 애플리케이션을 구현하기 위한 **상태 관리 패턴, 라이브러리**
- 여러 개의 컴포넌트 간의 데이터 관리를 쉽게 도와준다

Vuex?

- 복잡한 애플리케이션을 구현하기 위한 **상태 관리 패턴, 라이브러리**
- 여러 개의 컴포넌트 간의 데이터 관리를 쉽게 도와준다
- MVC 패턴에서 생기는 문제점을 보완하는 개발 패턴

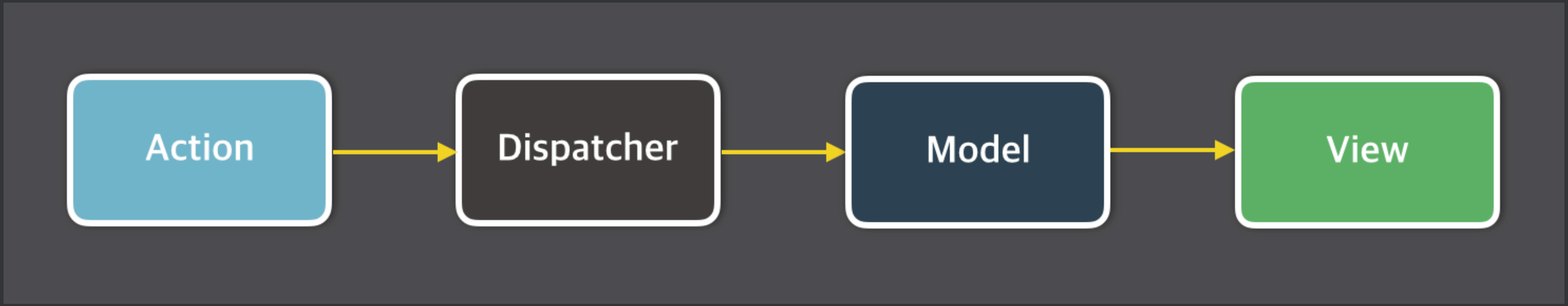
Flux

Flux

- Unidirectional Data Flow

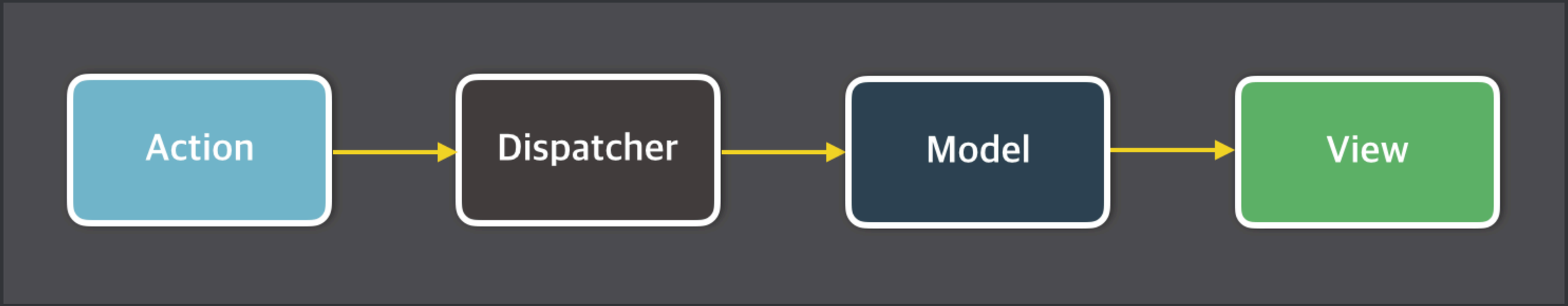
Flux

- Unidirectional Data Flow

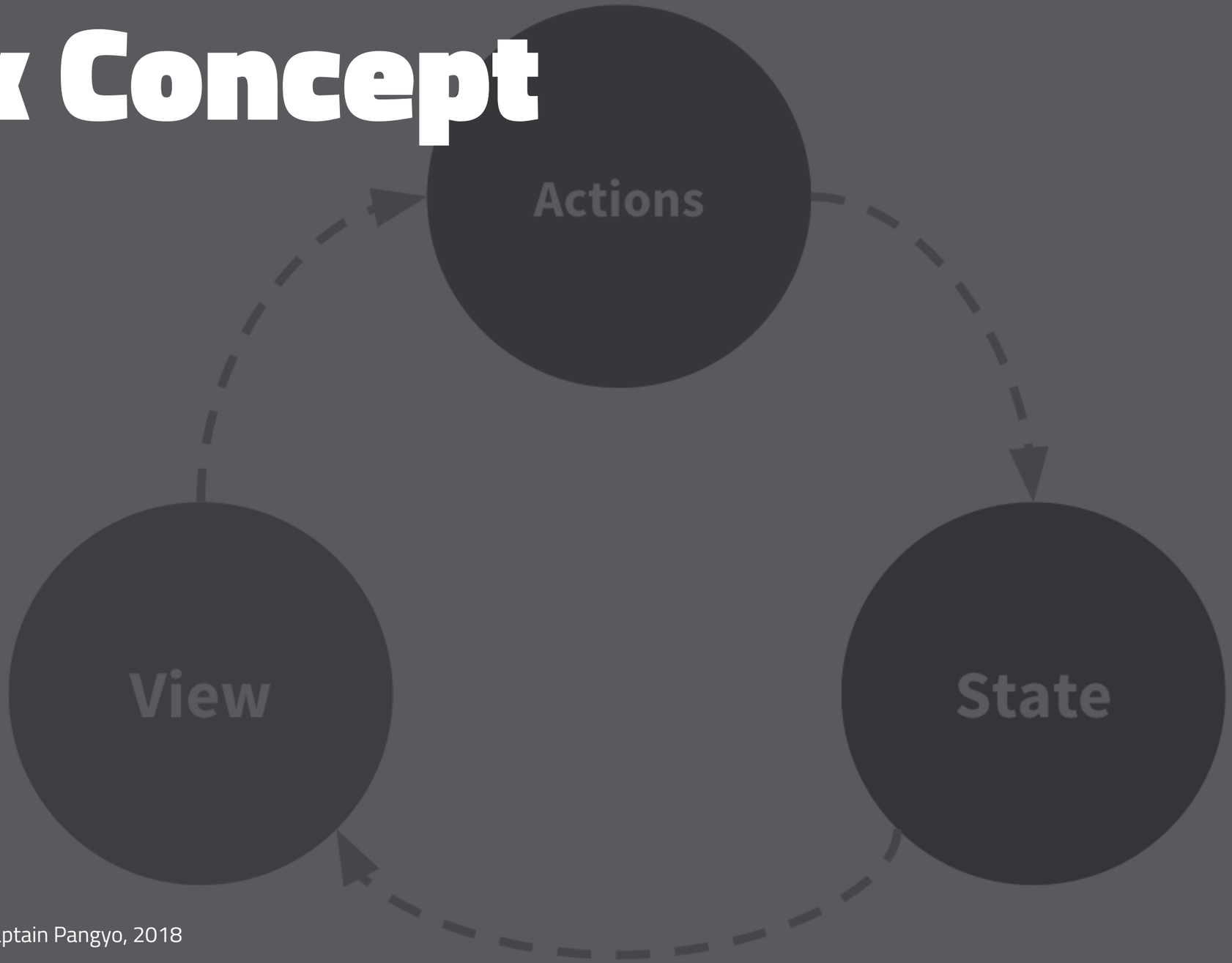


Flux

- Unidirectional Data Flow

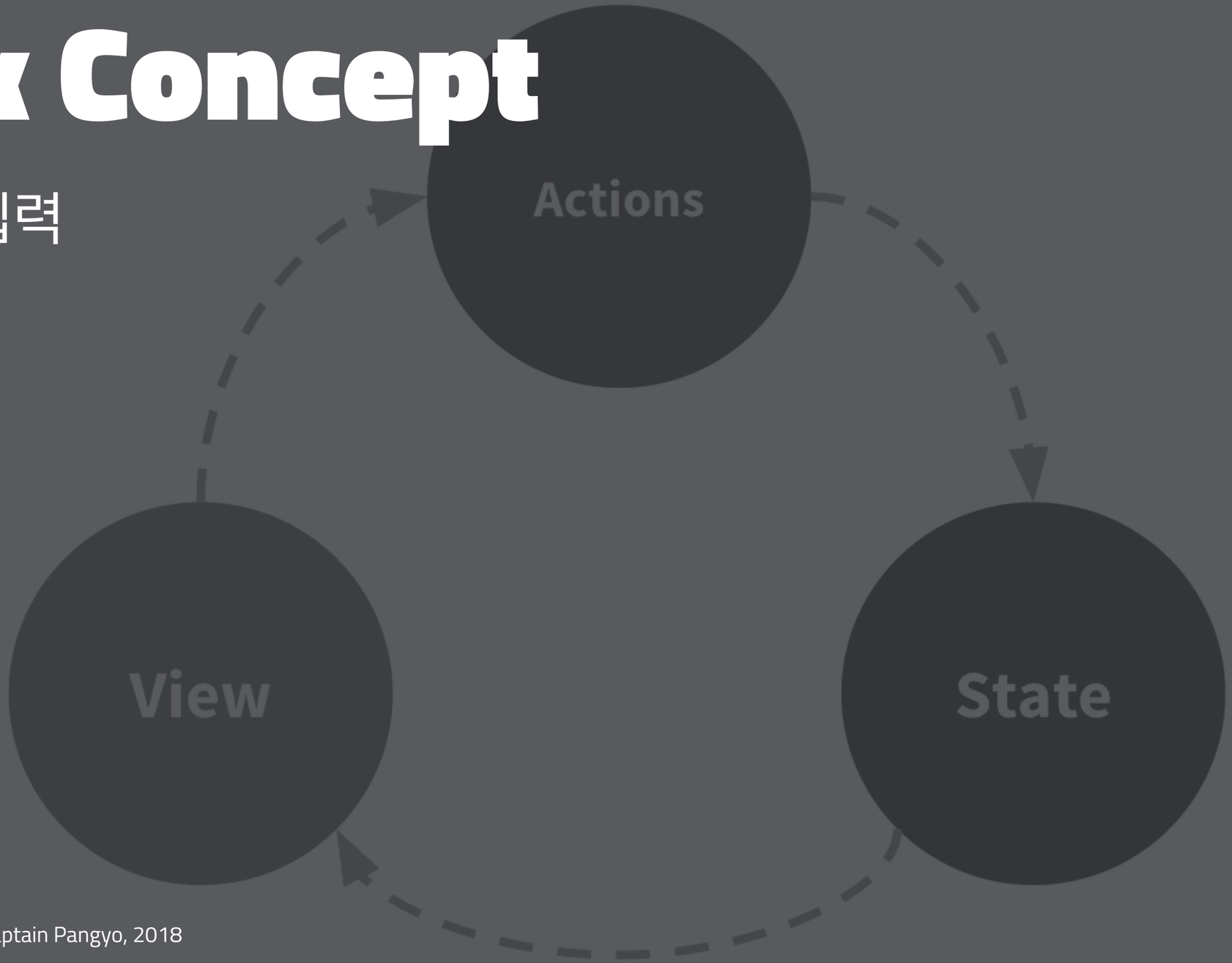


Vuex Concept



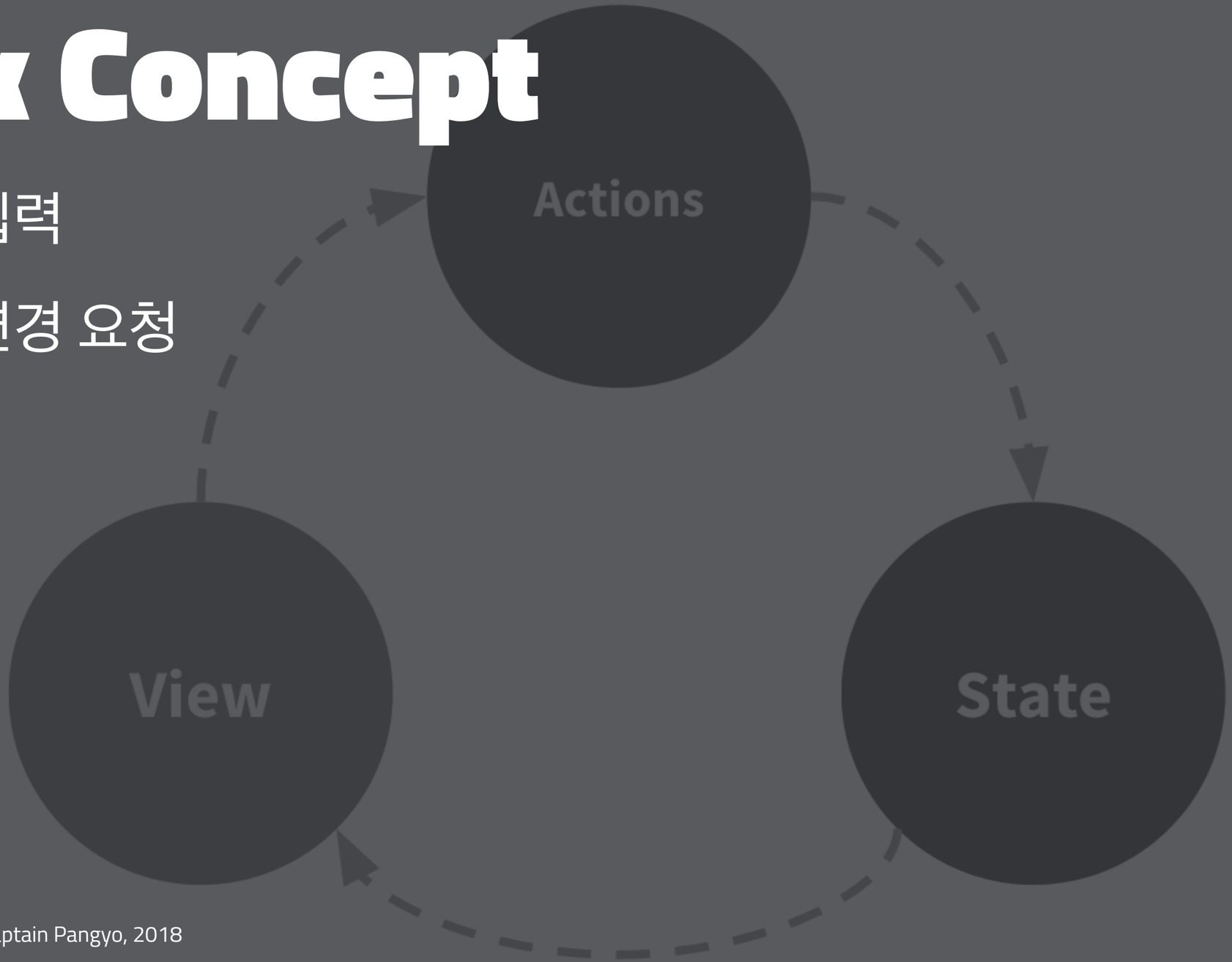
Vuex Concept

1. 사용자 입력



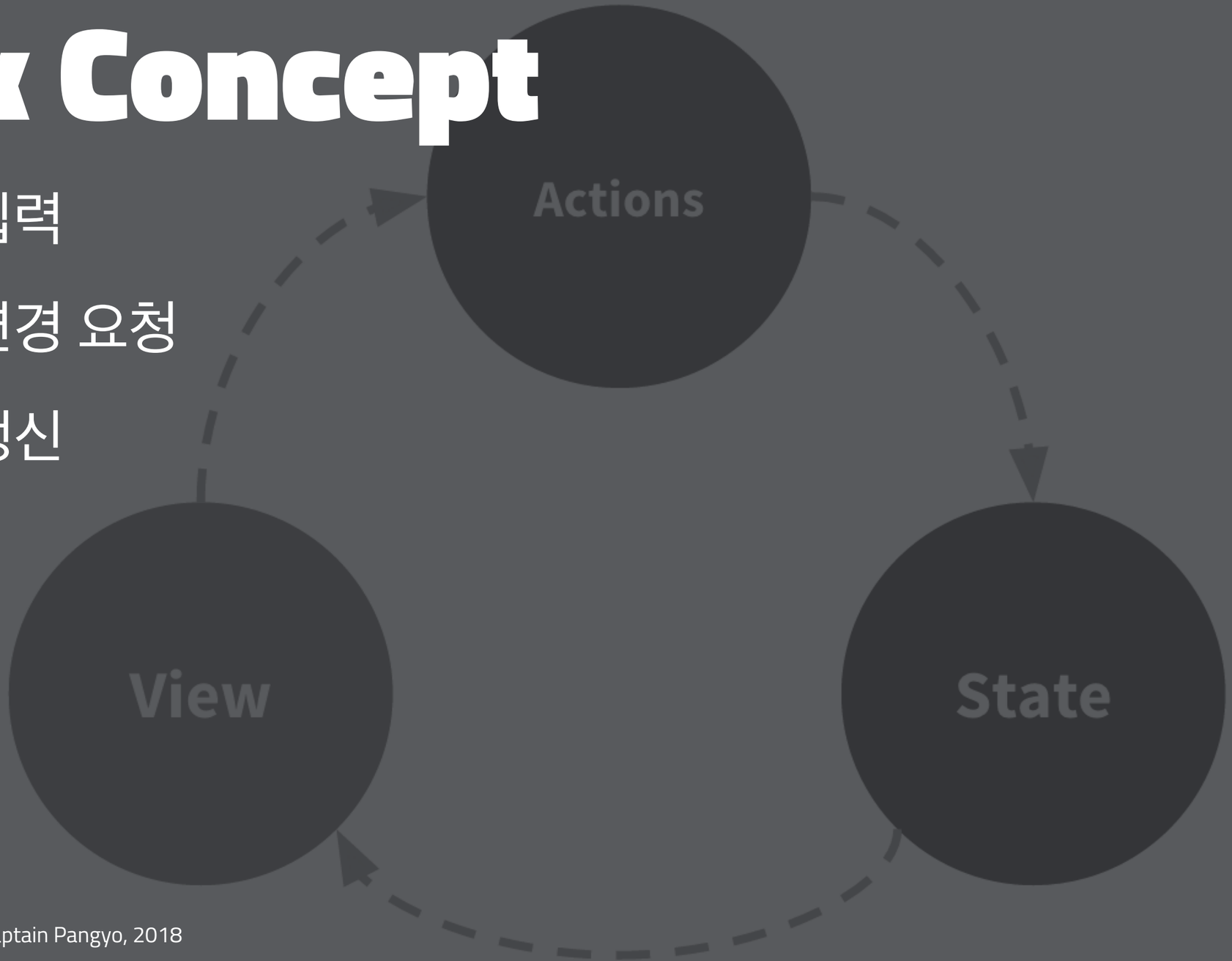
Vuex Concept

1. 사용자 입력
2. 데이터 변경 요청

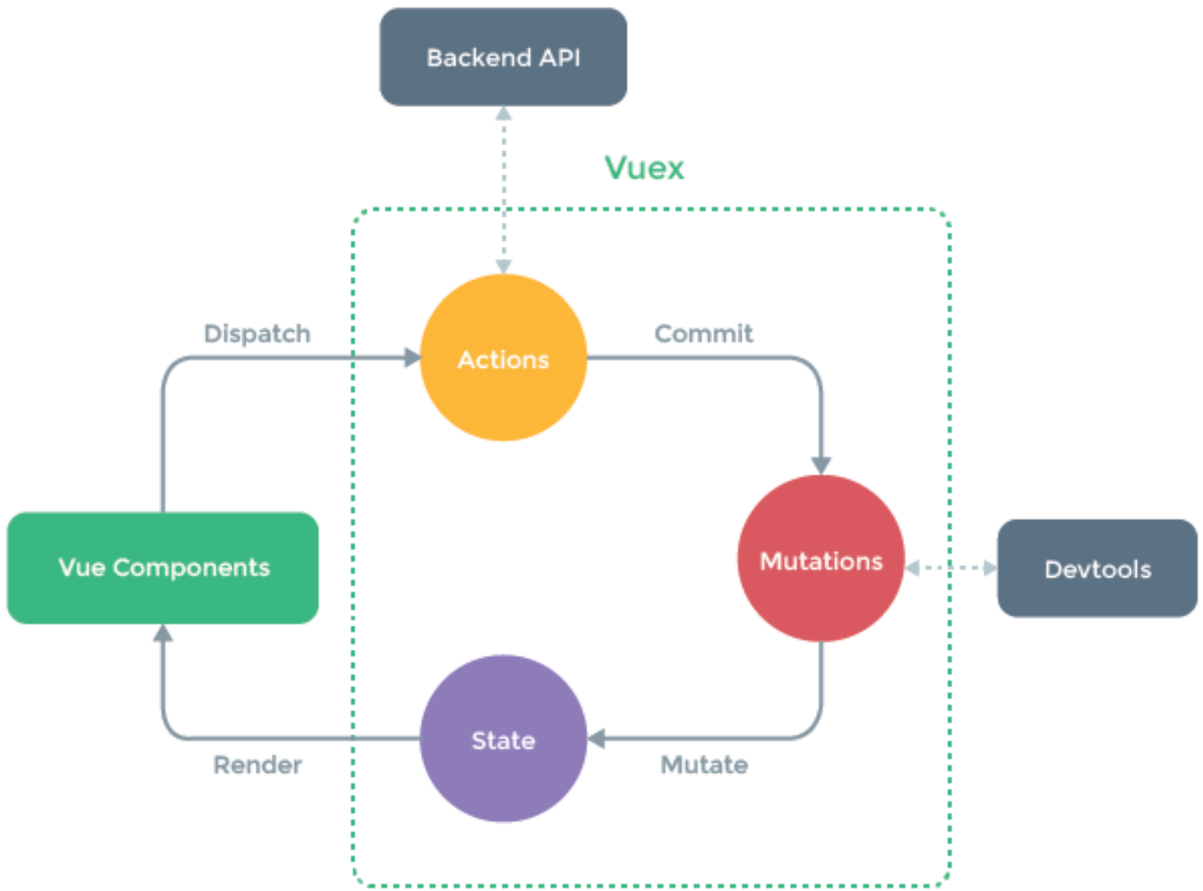


Vuex Concept

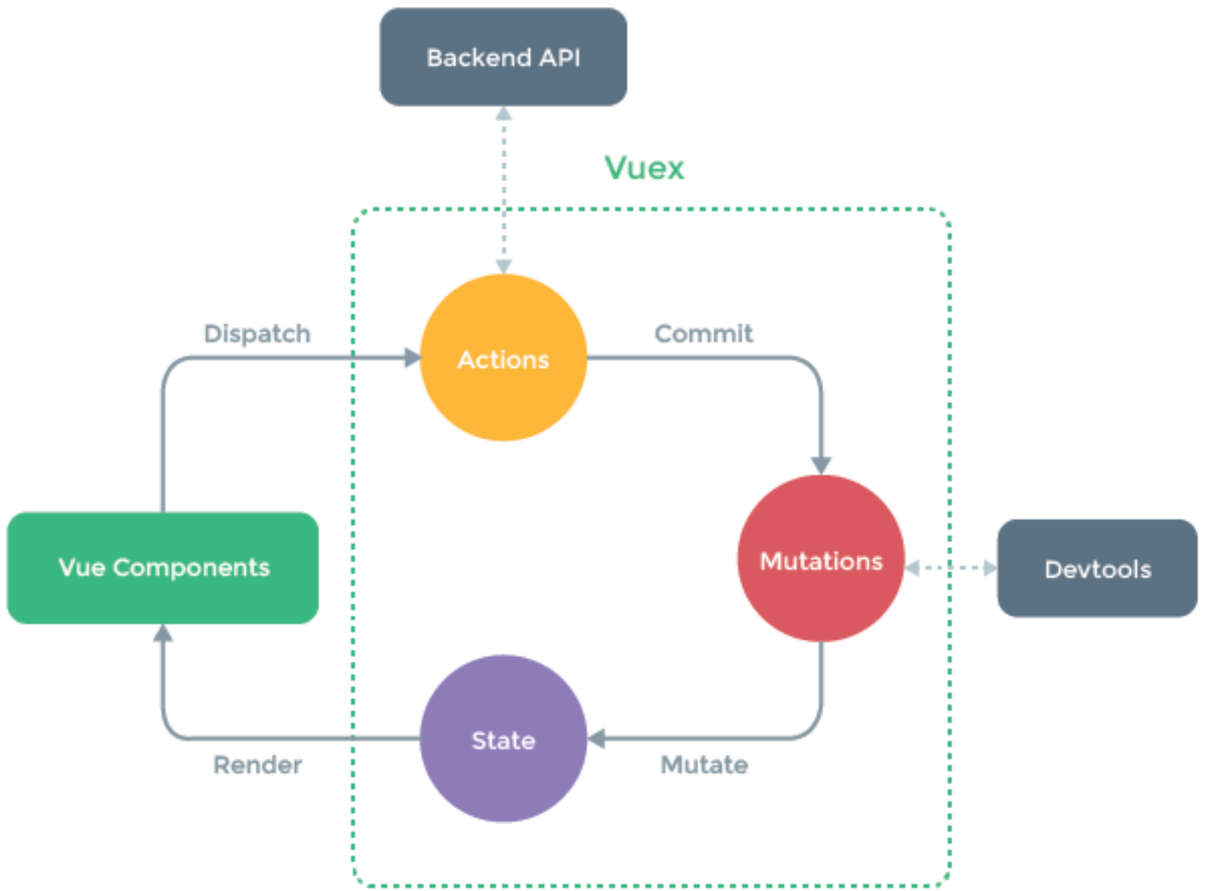
1. 사용자 입력
2. 데이터 변경 요청
3. 데이터 갱신



Vuex Structure

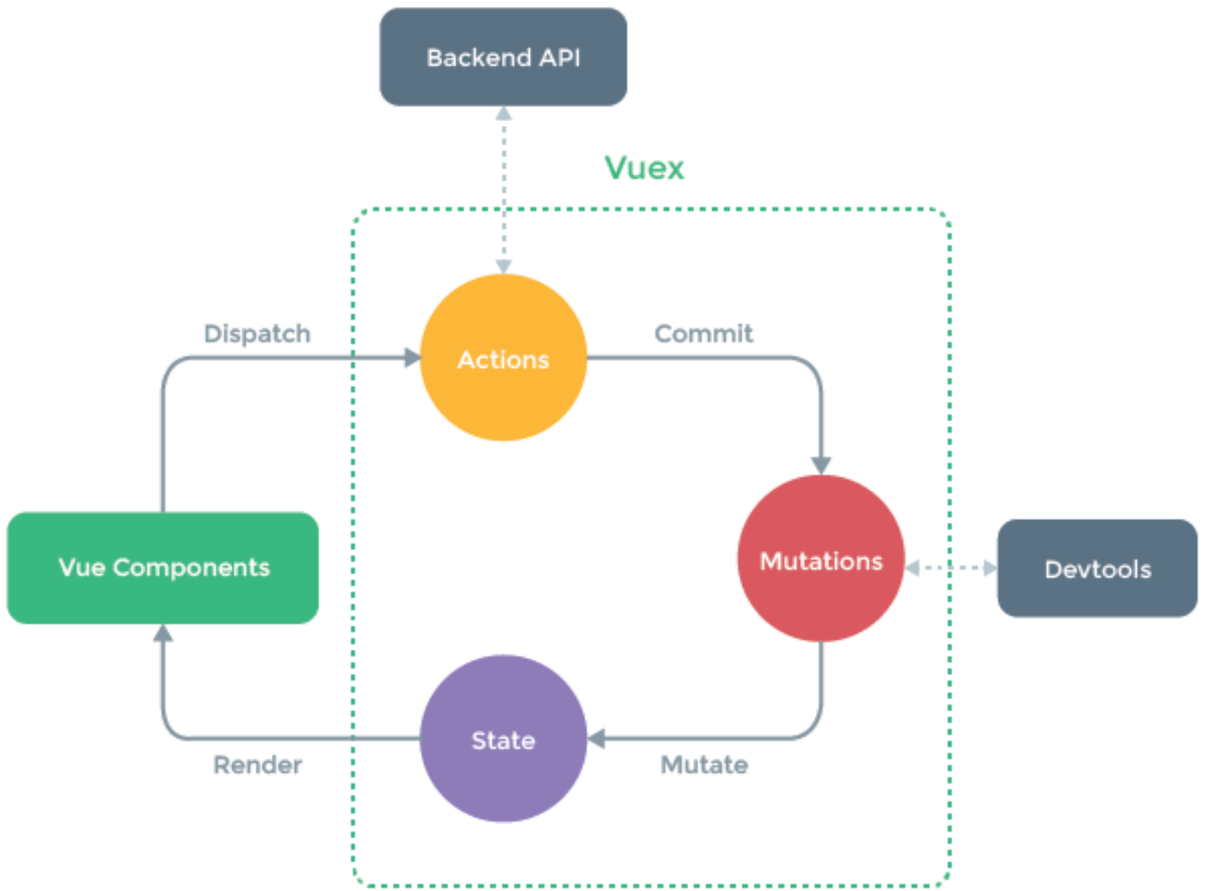


Vuex Structure



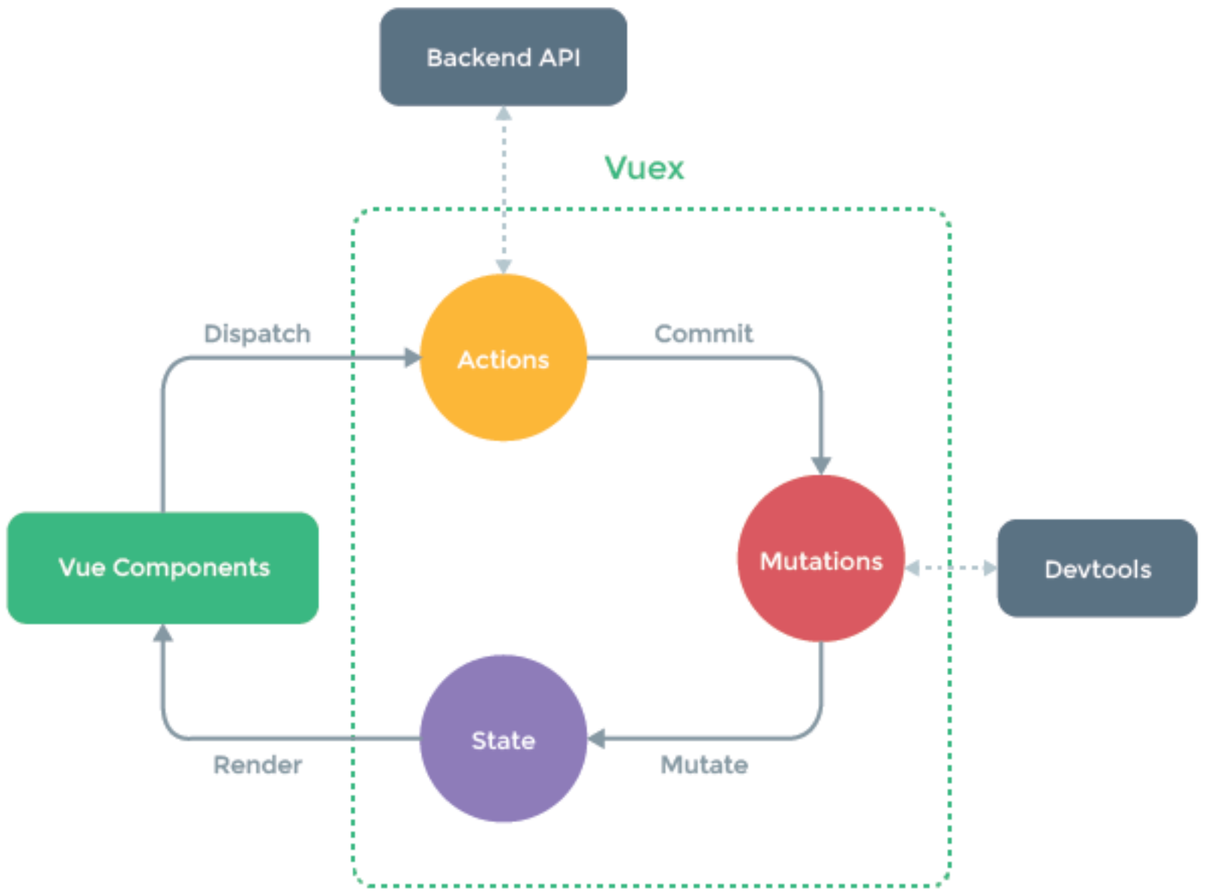
- State : 여러 컴포넌트 간에 **공유되는 데이터**

Vuex Structure



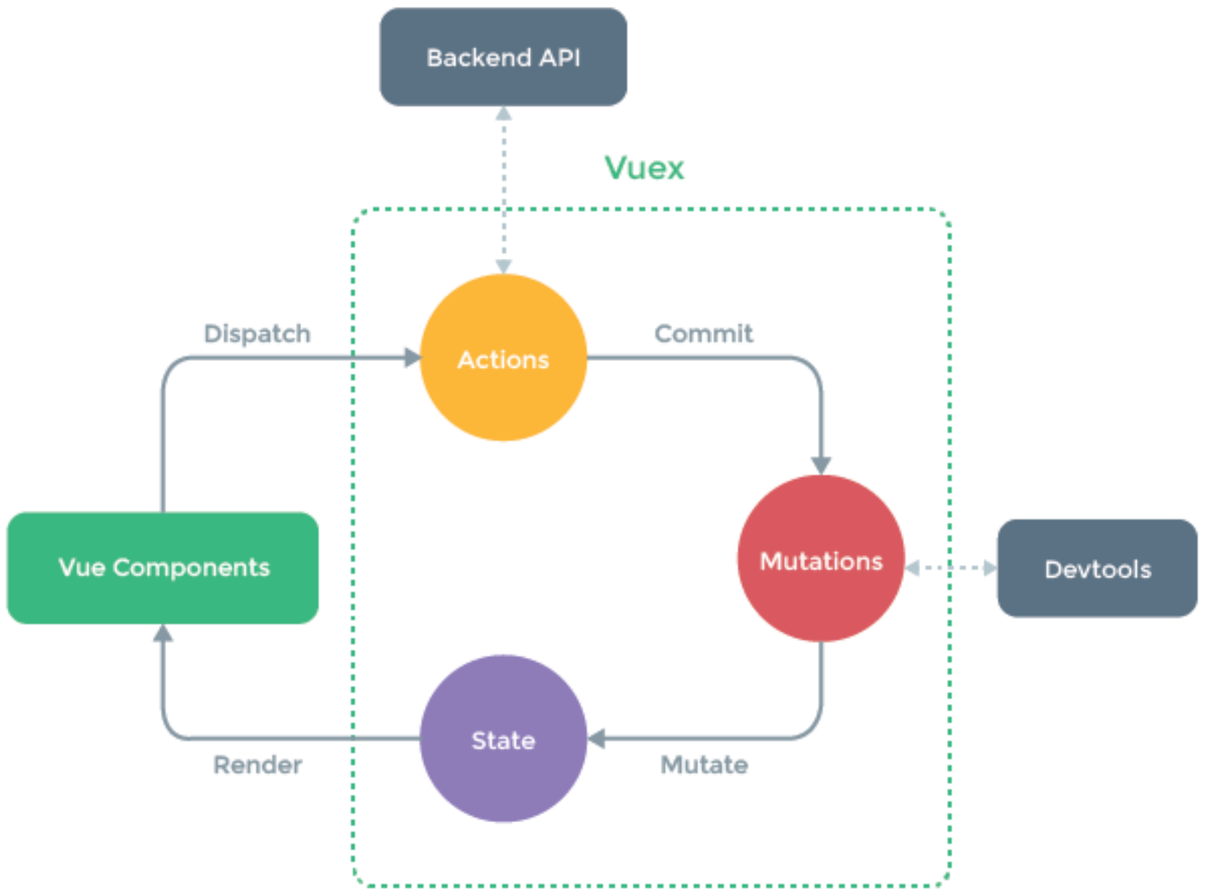
- State : 여러 컴포넌트 간에 **공유되는 데이터**
- Mutations : state를 **갱신**하는 로직

Vuex Structure



- State : 여러 컴포넌트 간에 **공유되는 데이터**
- Mutations : state를 **갱신**하는 로직
- Actions : 데이터 요청 및 **비동기** 로직

Vuex Structure



- State : **`data()`**
- Mutations : **`methods()`**
- Actions : **`async methods()`**

State

```
// Vue
data: {
    message: ‘hello vuex’
}
```

```
<p>{{ message }}</p>
```

State

```
// Vuex
state: {
    message: ‘hello vuex’
}
```

```
<p>{{ this.$store.state.message }}</p>
```

Getters

```
// Vue
computed: {
    doubleNum() {
        return this.num * 2;
    }
}
```

```
<p>{{ doubleNum }}</p>
```

Getters

```
// Vuex
getters: {
    doubleNum(state) {
        return state.num * 2;
    }
}
```

```
<p>{{ this.$store.getters.doubleNum }}</p>
```

Mutations

```
// Vue
methods: {
    addNumber(inputNum) {
        this.num = this.num + inputNum;
    }
}
```

```
<button @click="addNumber(10)">add</button>
```

Mutations

```
// Vuex
mutations: {
    addNumber(state) {
        state.num = state.num + inputNum;
    }
}

this.$store.commit('addNumber', 10);
```

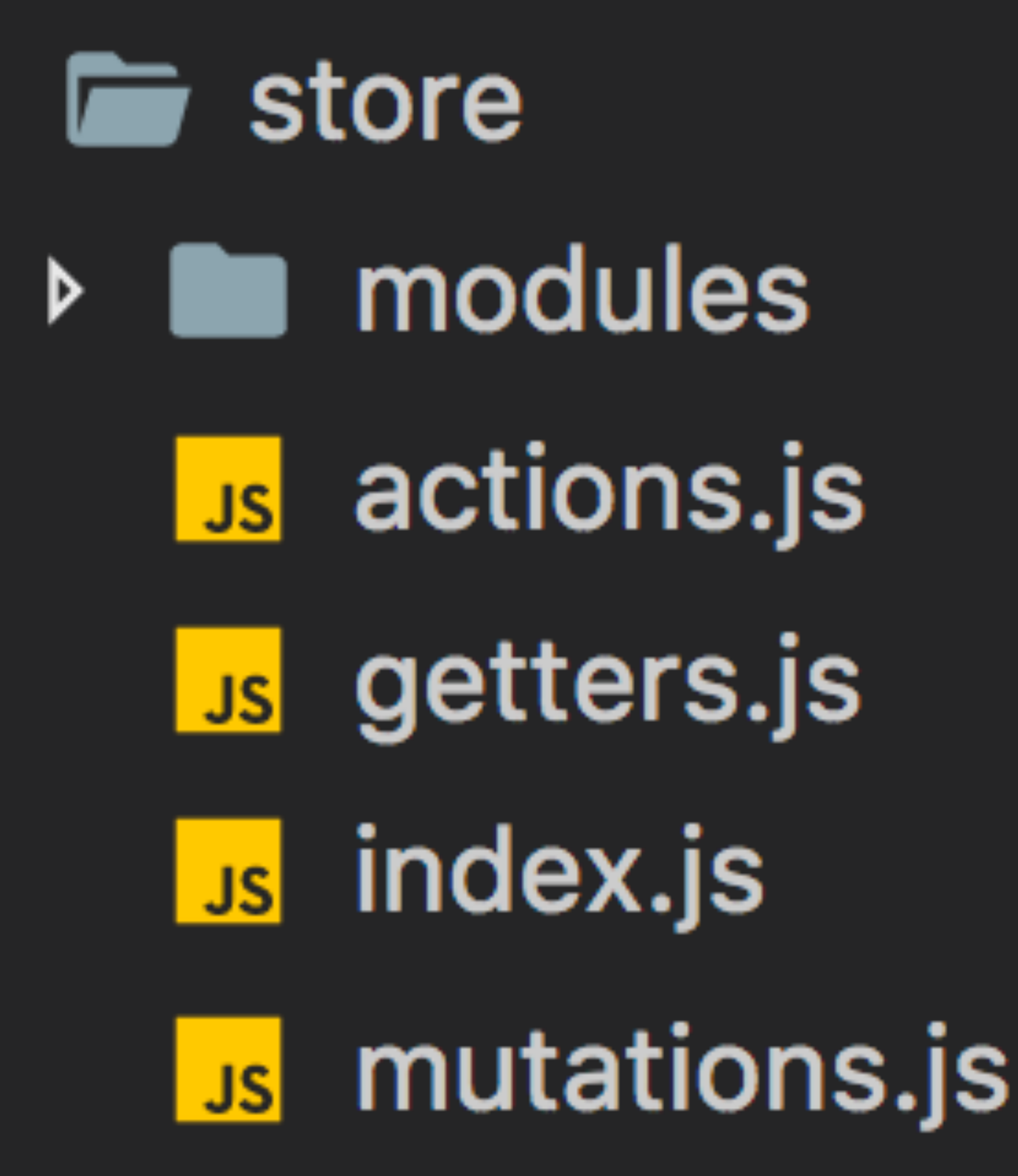
Actions

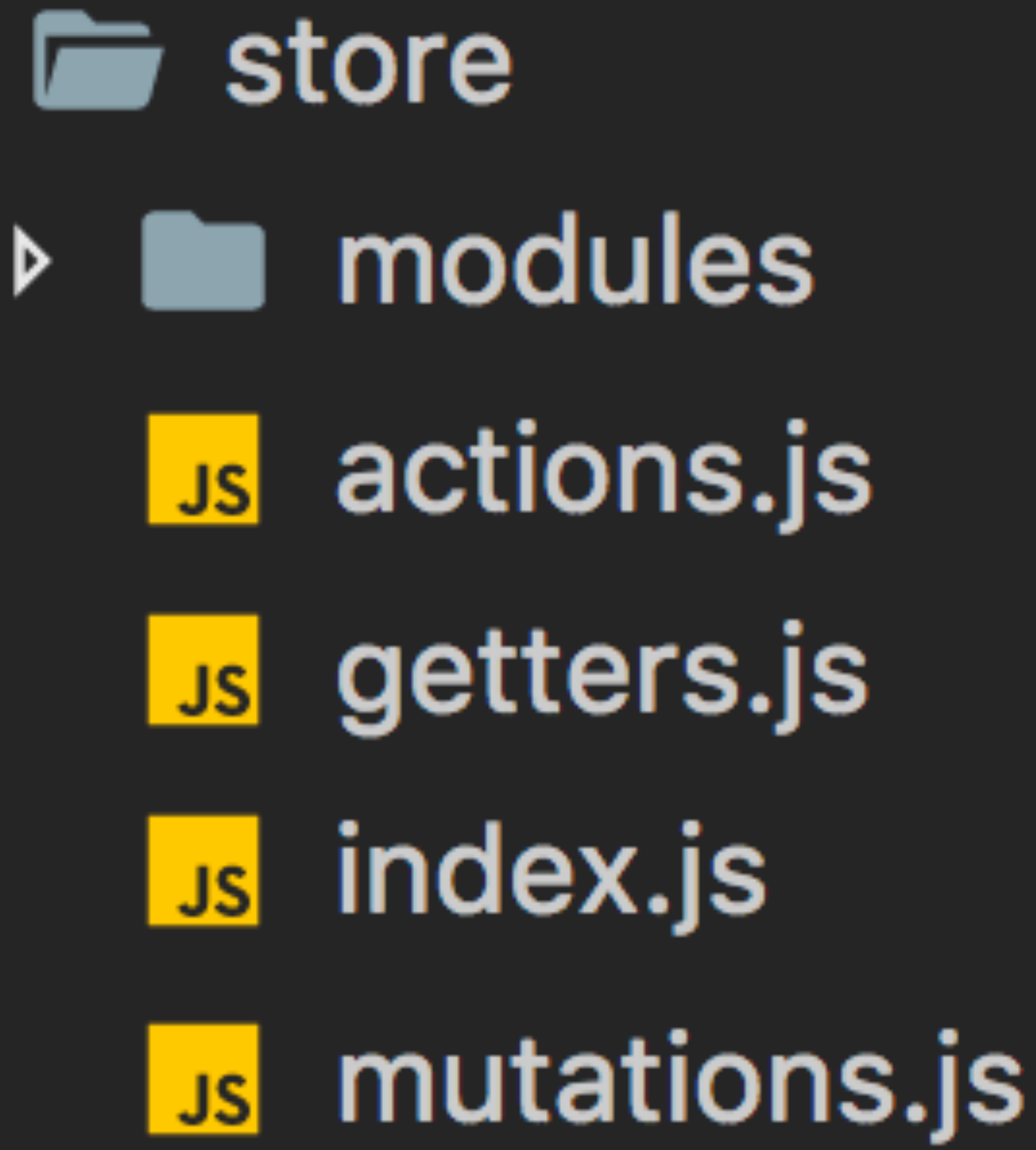
```
// Vue
methods: {
    fetchData() {
        return axios.get(‘items/1’)
            .then(res => item = res);
    }
}

<button @click=“fetchData”>get item</button>
```

Actions

```
// Vuex
actions: {
    fetchData({commit}) {
        return axios.get('items/1')
            .then(res => commit('setItem', res));
    }
}

this.$store.dispatch('fetchData');
```

Vuex Modularization





Vuex Modularization

```
import * as actions from ‘./actions’
import * as mutations from ‘./mutations’
import * as getters from ‘./getters’
import product from ‘./modules/product’

export default new Vuex.Store({
    state: {},
    getters,
    mutations,
    actions,
    modules: {
        product
    }
})
```

Vuex Coding Tip

```
import auth from ‘./modules/auth’
import posts from ‘./modules/posts’
import comments from ‘./modules/comments’
// …

export default new Vuex.Store({
    modules: {
        auth,
        posts,
        comments
        // …
    }
})
```

Vuex Coding Tip

```
// modules.js
import camelCase from 'lodash/camelCase'
const requireModule = require.context('.', false, /\.js$/)
const modules = {}

requireModule.keys().forEach(fileName => {
  // exclude the index.js
  if (fileName === './index.js') return
  const moduleName = camelCase(
    fileName.replace(/(\.\/|\.js)/g, '')
  )
  modules[moduleName] = requireModule(fileName)
})
export default modules
```

Vuex Coding Tip

```
// modules.js
import camelCase from 'lodash/camelCase'
const requireModule = require.context('.', false, /\.js$/)
const modules = {}

requireModule.keys().forEach(fileName => {
  // exclude the index.js
  if (fileName === './index.js') return
  const moduleName = camelCase(
    fileName.replace(/(\.\/|\.js)/g, '')
  )
  modules[moduleName] = requireModule(fileName)
})
export default modules
```

Vuex Coding Tip

```
// modules.js
import camelCase from 'lodash/camelCase'
const requireModule = require.context('.', false, /\.js$/)
const modules = {}

requireModule.keys().forEach(fileName => {
  // exclude the index.js
  if (fileName === './index.js') return
  const moduleName = camelCase(
    fileName.replace(/(\.\/|\.js)/g, '')
  )
  modules[moduleName] = requireModule(fileName)
})
export default modules
```

Vuex Coding Tip

```
// modules.js
import camelCase from 'lodash/camelCase'
const requireModule = require.context('.', false, /\.js$/)
const modules = {}

requireModule.keys().forEach(fileName => {
  // exclude the index.js
  if (fileName === './index.js') return
  const moduleName = camelCase(
    fileName.replace(/(\.\/|\.js)/g, '')
  )
  modules[moduleName] = {
    // scope isolation
    namespaced: true,
    ...requireModule(filename)
  }
})
export default modules
```

Vuex Coding Tip

```
// modules.js
import camelCase from 'lodash/camelCase'
const requireModule = require.context('.', false, /\.js$/)
const modules = {}

requireModule.keys().forEach(fileName => {
  // exclude the index.js
  if (fileName === './index.js') return
  const moduleName = camelCase(
    fileName.replace(/(\.\/|\.js)/g, '')
  )
  modules[moduleName] = {
    // scope isolation
    namespaced: true,
    ...requireModule(filename)
  }
})
export default modules
```

Vuex Coding Tip

```
import auth from ‘./modules/auth’
import posts from ‘./modules/posts’
import comments from ‘./modules/comments’
// …

export default new Vuex.Store({
    modules: {
        auth,
        posts,
        comments
        // …
    }
})
```

Vuex Coding Tip

```
import modules from ‘./modules’

export default new Vuex.Store({
    modules
})
```

Don’t do in Vuex

Don't do in Vuex

- all data properties in **`Store`** (x)

Don’t do in Vuex

- all data properties in **`Store`** (x)
- underestimate **`eventBus`** (x)

3. Proven Patterns

Smarter Watchers

Smarter Watchers

```
created() {
    this.fetchUserList()
},
watch: {
    searchText() {
        this.fetchUserList()
    }
}
```

Smarter Watchers

```
created() {
    this.fetchUserList()
},
watch: {
    searchText: ‘fetchUserList’
}
```

Smarter Watchers

```
created() {
    this.fetchUserList()
},
watch: {
    searchText: ‘fetchUserList’ // accept methods name
}
```

Smarter Watchers

```
created() { … },
watch: {
    searchText: {
        handler: ‘fetchUserList’
    }
}
```

Smarter Watchers

```
// created() { … },
watch: {
    searchText: {
        handler: ‘fetchUserList’,
        immediate: true // Bye!! created()
    }
}
```

Component Registration

Component Registration

```
import BaseButton from ‘./base-button’
import BaseIcon from ‘./base-icon’
import BaseInput from ‘./base-input’

export default {
    components: {
        BaseButton,
        BaseIcon,
        BaseInput
    }
}
```

```
import BaseButton from ‘./base-button’
import BaseIcon from ‘./base-icon’
import BaseInput from ‘./base-input’

export default {
    components: {
        BaseButton,
        BaseIcon,
        BaseInput
    }
}
```

Too frequent!!

Component Registration

```
// components/_globals.js
import Vue from 'vue'
import upperFirst from 'lodash/upperFirst'
import camelCase from 'lodash/camelCase'

// Require in a base component context
const requireComponent = require.context(
  '.', false, /base-[\w-]+\.vue$/
)

requireComponent.keys().forEach(fileName => {
  // Get component config
  const componentConfig = requireComponent(fileName)
  // Get PascalCase name of component
  const componentName = upperFirst(
    camelCase(fileName.replace(/^\.\//, '').replace(/\.\w+$/, ''))
  )
  // Register component globally
  Vue.component(componentName, componentConfig.default || componentConfig)
})
```

Component Registration

```
// components/_globals.js
import Vue from 'vue'
import upperFirst from 'lodash/upperFirst'
import camelCase from 'lodash/camelCase'

// Require in a base component context
const requireComponent = require.context(
  '.', false, /base-[\w-]+\.vue$/
)

requireComponent.keys().forEach(fileName => {
  // Get component config
  const componentConfig = requireComponent(fileName)
  // Get PascalCase name of component
  const componentName = upperFirst(
    camelCase(fileName.replace(/^\.\//, '').replace(/\.\w+$/, ''))
  )
  // Register component globally
  Vue.component(componentName, componentConfig.default || componentConfig)
})
```

Component Registration

```
// components/_globals.js
import Vue from 'vue'
import upperFirst from 'lodash/upperFirst'
import camelCase from 'lodash/camelCase'

// Require in a base component context
const requireComponent = require.context(
  '.', false, /base-[\w-]+\.vue$/
)

requireComponent.keys().forEach(fileName => {
  // Get component config
  const componentConfig = requireComponent(fileName)
  // Get PascalCase name of component
  const componentName = upperFirst(
    camelCase(fileName.replace(/^\.\//, '').replace(/\.\w+$/, ''))
  )
  // Register component globally
  Vue.component(componentName, componentConfig.default || componentConfig)
})
```

Component Registration

```
// components/_globals.js
import Vue from 'vue'
import upperFirst from 'lodash/upperFirst'
import camelCase from 'lodash/camelCase'

// Require in a base component context
const requireComponent = require.context(
  '.', false, /base-[\w-]+\.vue$/
)

requireComponent.keys().forEach(fileName => {
  // Get component config
  const componentConfig = requireComponent(fileName)
  // Get PascalCase name of component
  const componentName = upperFirst(
    camelCase(fileName.replace(/^\.\//, '').replace(/\.\w+$/, ''))
  )
  // Register component globally
  Vue.component(componentName, componentConfig.default || componentConfig)
})
```

Multiple Root Elements

```
Error compiling template:

<div id="app">
  First Element
</div>
<div>
  Second Element
</div>

- Component template should contain exactly one root element.
```

Multiple Root Elements

```
<template>
    <ul>
        <NavBarRoutes :routes="persistentNavRoutes"/>
        <NavBarRoutes v-if="loggedIn" :routes="loggedInNavRoutes"/>
        <NavBarRoutes v-else :routes="loggedOutNavRoutes"/>
    </ul>
</template>
```

Multiple Root Elements

```
// NavBarRoutes 템플릿
<template>
  <li v-for="route in routes" :key="route.name”>
    <router-link :to="route">
      {{ route.title }}
    </router-link>
  </li>
</template>
```

Multiple Root Elements

```
// NavBarRoutes 템플릿
<template>
  <li v-for="route in routes" :key="route.name”>
    <router-link :to="route">
      {{ route.title }}
    </router-link>
  </li>
</template>
```

Multiple Root Elements

```
// NavBarRoutes 템플릿
<template>
  // Error : Multi Root Elements!!
  <li v-for="route in routes" :key="route.name”>
    <router-link :to="route">
      {{ route.title }}
    </router-link>
  </li>
</template>
```

Multiple Root Elements

```
functional: true,
render(h, { props }) {
  return props.routes.map(route =>
    <li key={route.name}>
      <router-link to={route}>
        {route.title}
      </router-link>
    </li>
  )
}
```

4. Real World Examples

Using External Libraries

Using External Libraries

- 프런트엔드 화면 라이브러리 **`Bootstrap..`**

Using External Libraries

- 프런트엔드 화면 라이브러리 **`Bootstrap..`**
- 데이터 시각화를 위한 차트 라이브러리 **`HighChart..`**

Bootstrap + Vue

Build responsive, mobile-first projects on the web using Vue.js and the world's most popular front-end CSS library — Bootstrap V4.

Bootstrap-Vue provides one of the most comprehensive implementations of Bootstrap V4 components and grid system available for Vue.js 2.4+, complete with extensive and automated WAI-ARIA accessibility markup.

Bootstrap 4 is the world's most popular framework for building responsive, mobile-first sites.

Vue.js (pronounced /vjuË, like view) is a progressive framework for building user interfaces.

Get started | Github

Currentiy v2.0.0-rc.8

Bootstrap Integration

```
// cli
npm i bootstrap jquery popper

// main.js
import 'bootstrap'
import 'bootstrap/dist/css/bootstrap.min.css'

// webpack.config.js
plugins: [
  new webpack.ProvidePlugin({
    $: "jquery"
  })
],
```

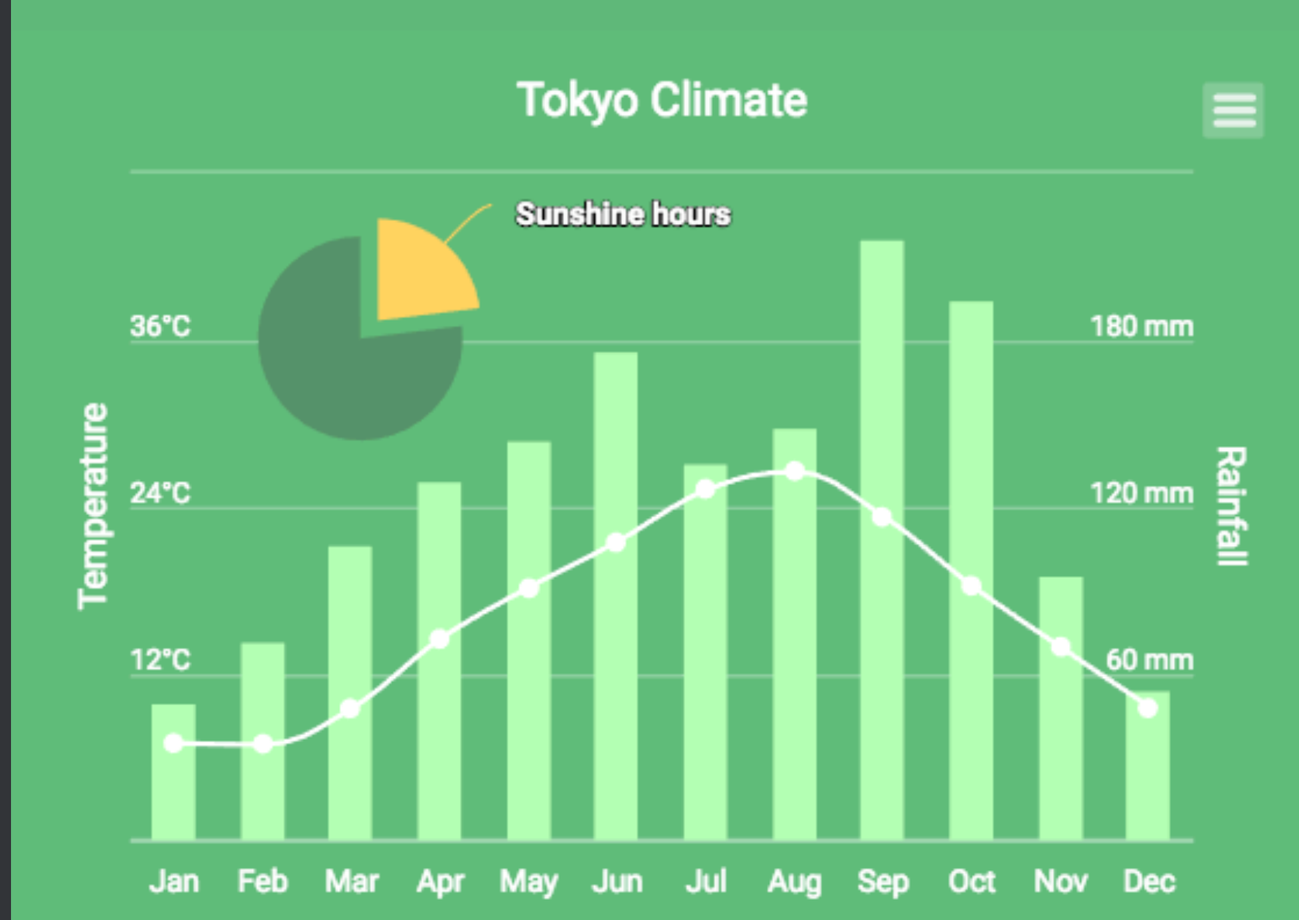
Bootstrap Integration

```
// bootstrap-init.js
// 1. popover
$(function () {
  $('[data-toggle="popover"]').popover()
})

// 2. tooltip
$(function () {
  $('[data-toggle="tooltip"]').tooltip()
})
…
```

```
<button type="button" data-toggle="tooltip" data-placement="top">
  Tooltip on top
</button>
```

Library Modularization

- **install** library
- **import** library
- **modularize** chart component
- use the library as **plugin**



Highcharts ›

Create interactive charts easily for your web projects.

Used by tens of thousands of developers and 72 out of the world's 100 largest companies, Highcharts is the simplest yet most flexible charting API on the market.

Install and Import

```
// cli
npm install highcharts --save

// App.vue
import Highcharts from ‘highcharts’

export default {
  mounted() {
    Highcharts.chart('container', { /* Highcharts options */ });
  }
}

<div id="container" style="width:100%; height:400px;"></div>
```

Component Modularization

```
// BarChart.vue
import Highcharts from ‘highcharts’

export default {
  mounted() {
    Highcharts.chart('container', { /* Highcharts options */ });
  }
}

<div id="container" style="width:100%; height:400px;"></div>
```

Component Modularization

```
// App.vue
import BarChart from ‘./components/BarChart’

export default {
  components: {
    BarChart
  }
}

<div id=“app”>
  <BarChart></BarChart>
</div>
```

Chart as a plugin

```
import Vue from 'vue'
import App from './App.vue'
import HighChartPlugin
from "./plugins/highchart-plugins";

Vue.use(HighChartPlugin);

new Vue({
  el: '#app',
  render: h => h(App)
})
```

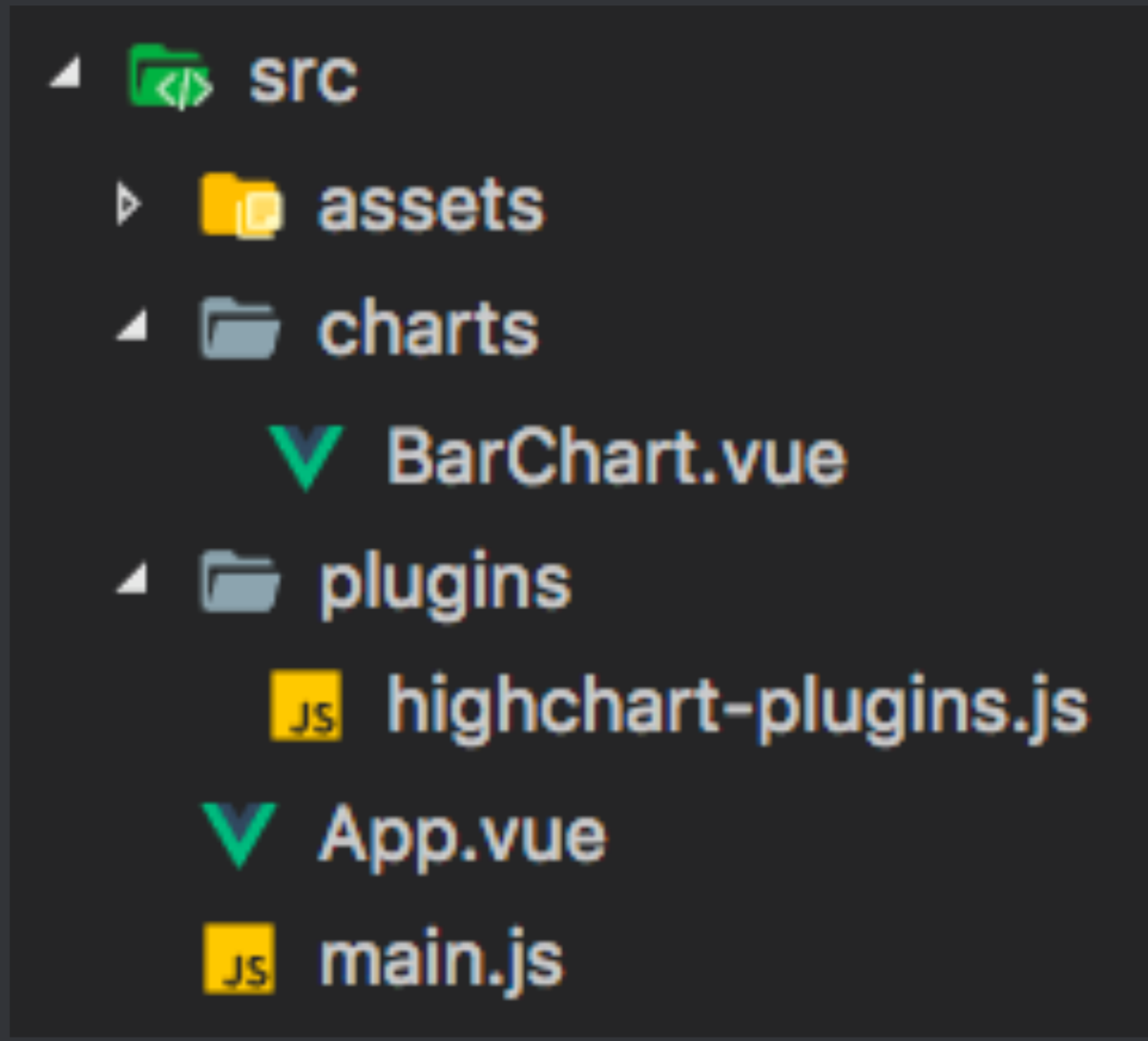


Chart as a plugin

```
import Vue from 'vue'
import App from './App.vue'
import HighChartPlugin
from "./plugins/highchart-plugins";

Vue.use(HighChartPlugin);

new Vue({
  el: '#app',
  render: h => h(App)
})
```

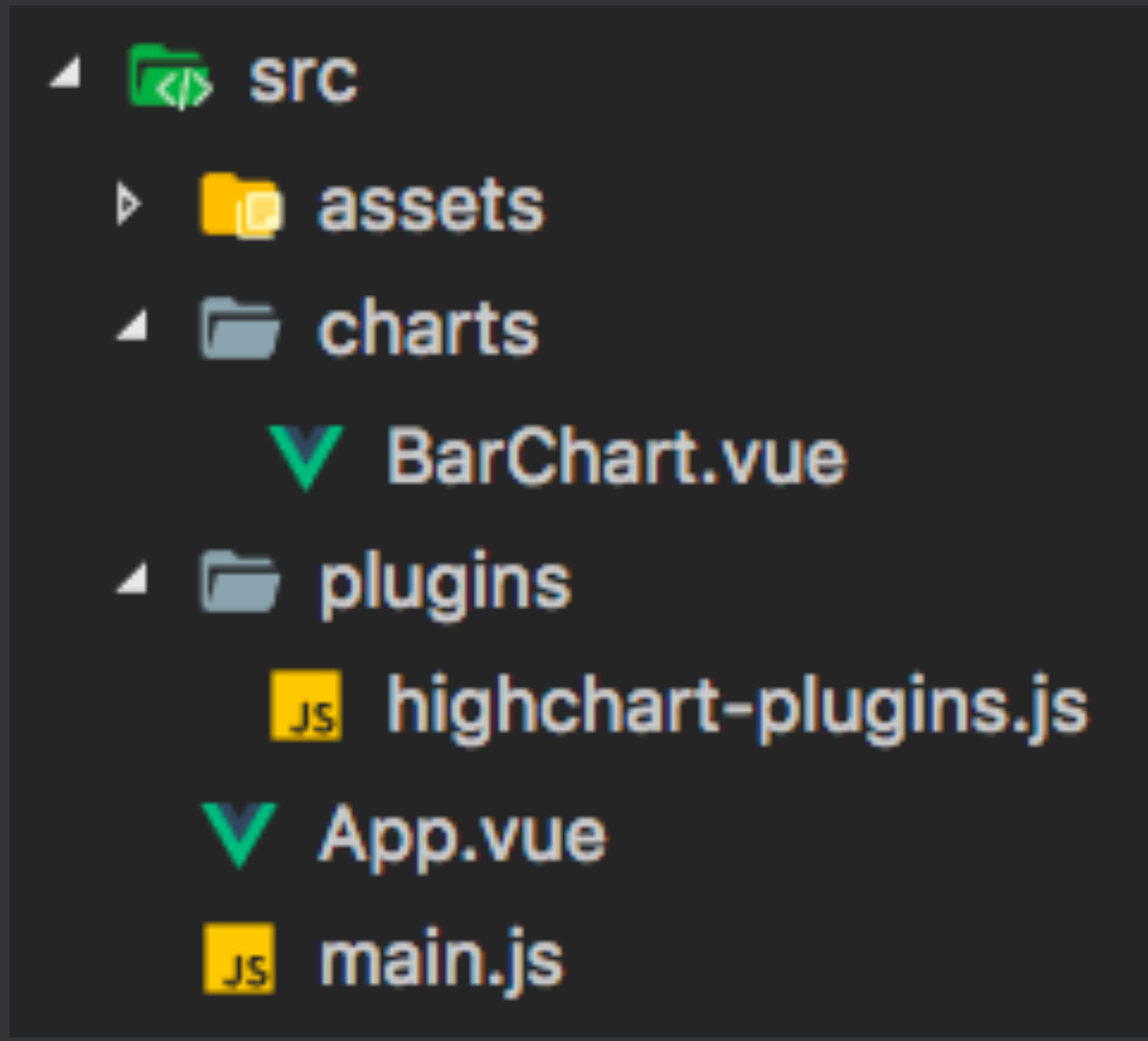


Chart as a plugin

```
import Highcharts from 'highcharts';

const HighChartPlugin = {
  install(Vue, options) {
    Vue.prototype.Highcharts = Highcharts;
  }
}

// Automatic installation if Vue has been added to the global scope.
if (typeof window !== 'undefined' && window.Vue) {
  window.Vue.use(MyPlugin)
}

export default HighChartPlugin;
```

Chart as a plugin

```
import Highcharts from 'highcharts';

const HighChartPlugin = {
  install(Vue, options) {
    Vue.prototype.Highcharts = Highcharts;
  }
}

// Automatic installation if Vue has been added to the global scope.
if (typeof window !== 'undefined' && window.Vue) {
  window.Vue.use(MyPlugin)
}

export default HighChartPlugin;
```

Chart as a plugin

```
// BarChart.vue
import Highcharts from ‘highcharts’

export default {
  mounted() {
    Highcharts.chart('container', { /* Highcharts options */ });
  }
}
```

Chart as a plugin

```
// BarChart.vue
// import Highcharts from ‘highcharts’

export default {
  mounted() {
    this.Highcharts.chart('container', { /* Highcharts options */ });
  }
}
```

Development Workflow (Vue + Spring)

Development Workflow (Vue + Spring)

Vue CLI

- Webpack Dev Server (localhost:8080)

Development Workflow (Vue + Spring)

Vue CLI

- Webpack Dev Server (localhost:8080)

Spring

- Web Application Server (localhost:8081)
- CORS Filter

Development Workflow (Vue + Node.js)

Development Workflow (Vue + Node.js)

Vue CLI

- Webpack Dev Middleware

Development Workflow (Vue + Node.js)

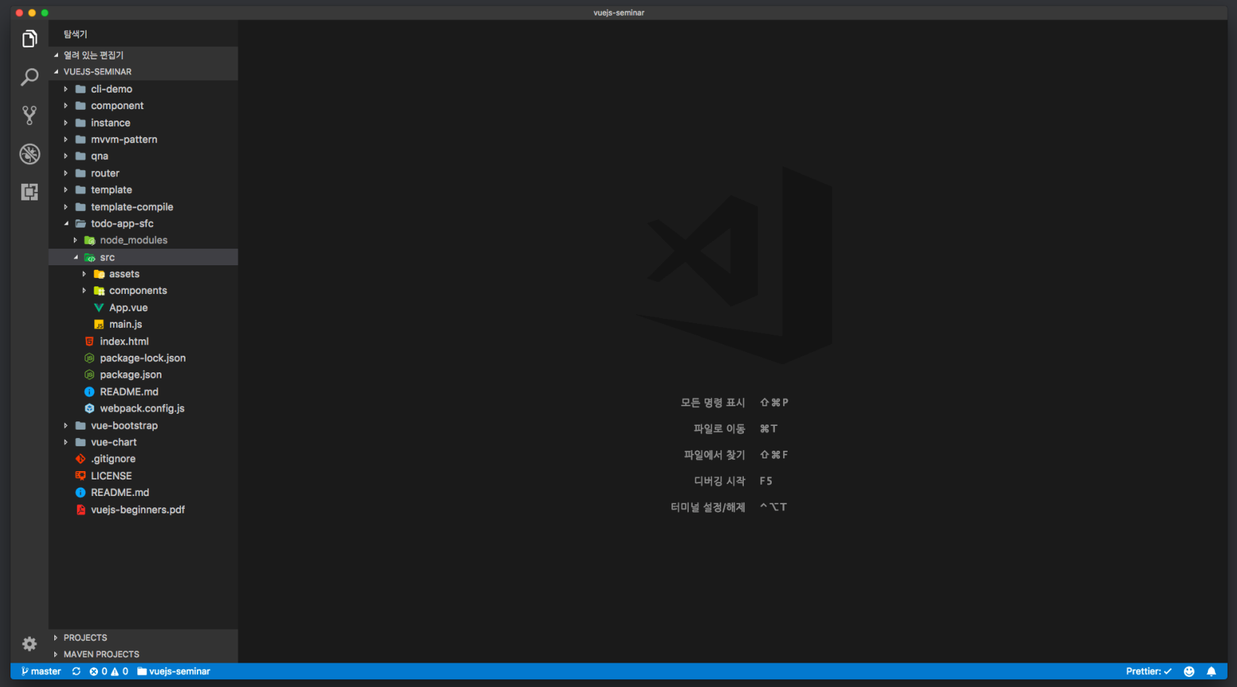
Vue CLI

- Webpack Dev Middleware

Node.js

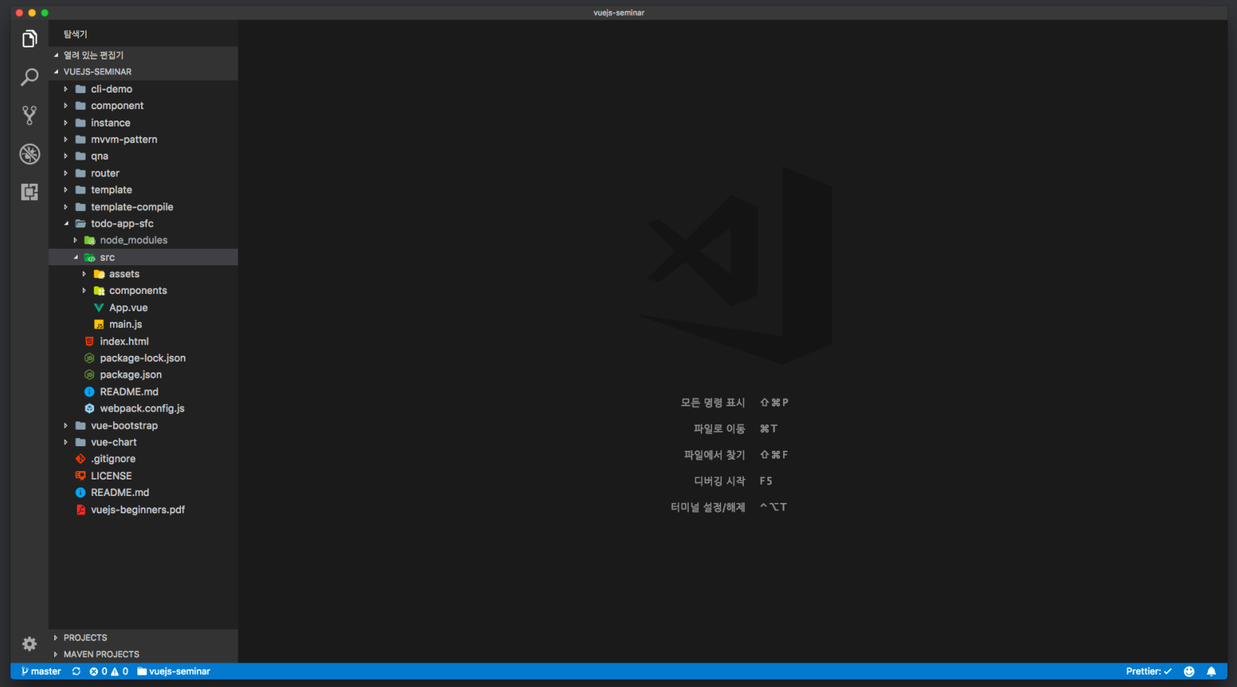
- Express (localhost:3000)
- Webpack

Useful Dev Tools



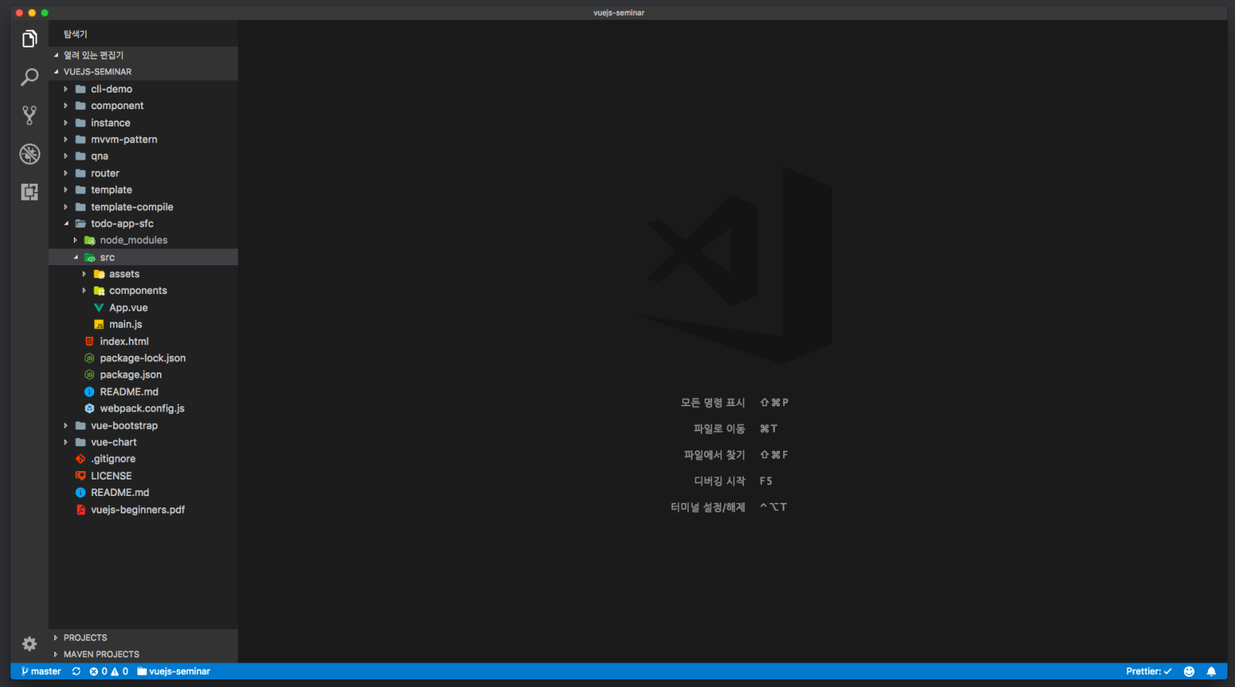
Useful Dev Tools

- VSCode
  - Vetur
  - TSlint
  - Prettier



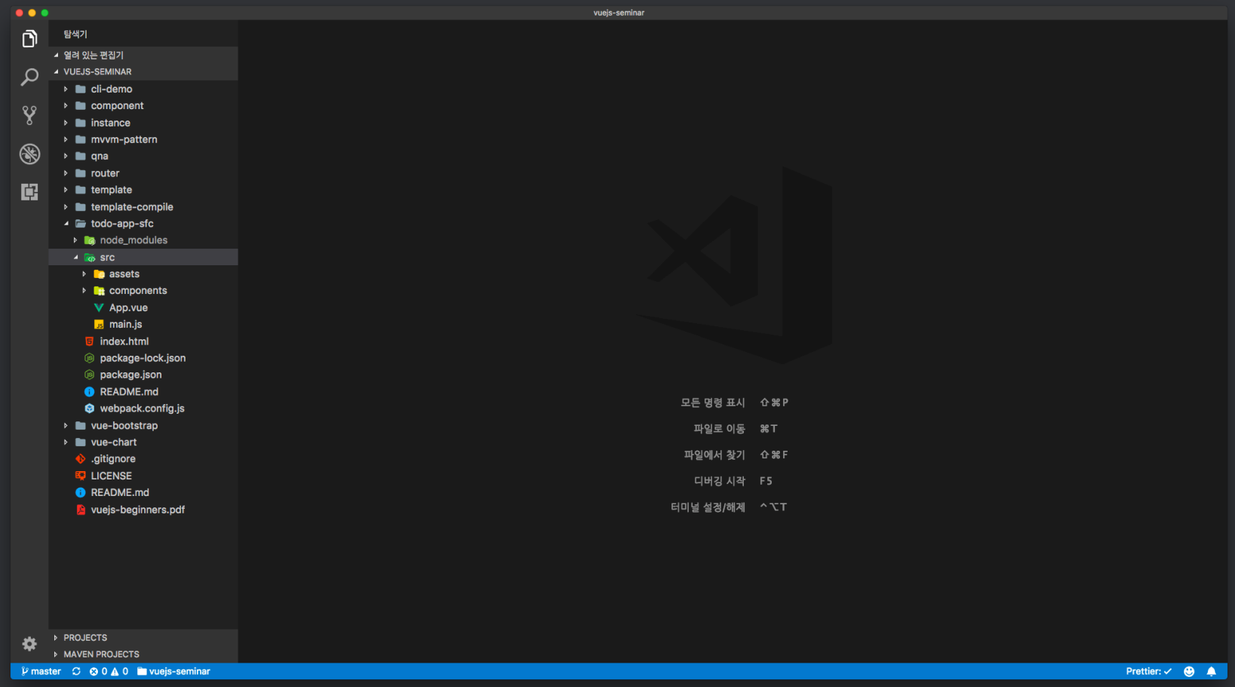
Useful Dev Tools

- VSCode
  - Vetur
  - TSlint
  - Prettier
- ATOM
  - language-vue



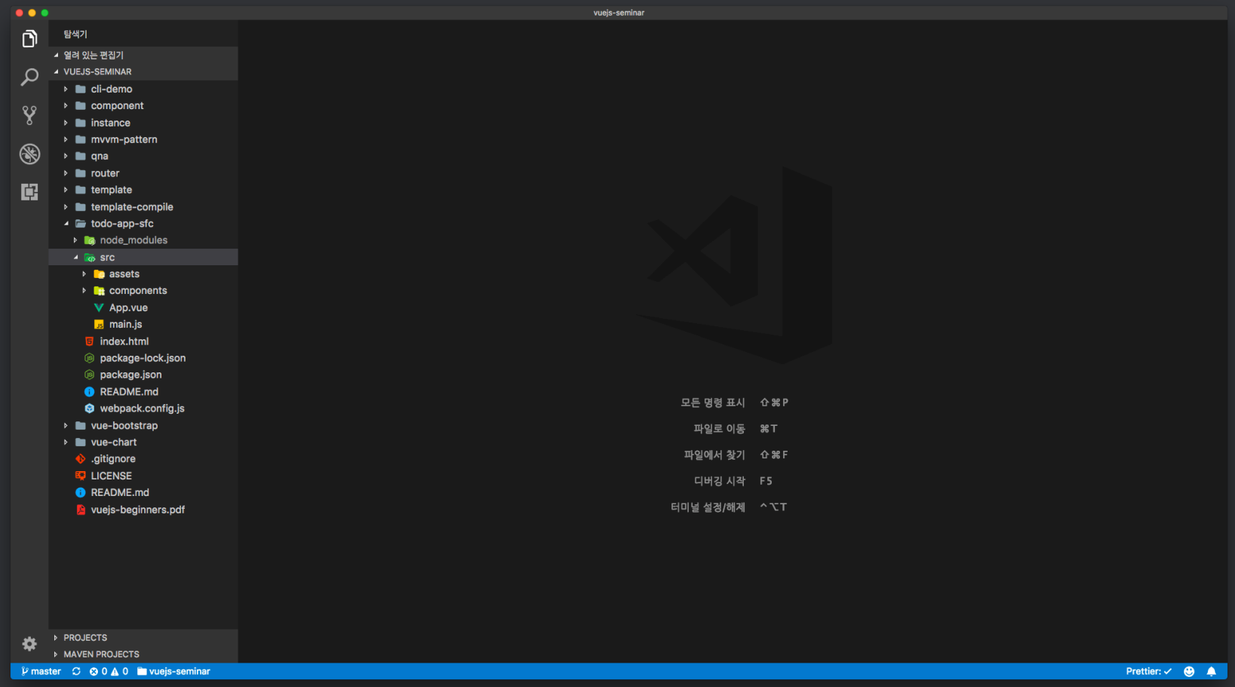
Useful Dev Tools

- VSCode
  - Vetur
  - TSlint
  - Prettier
- ATOM
  - language-vue
- Vue Dev Tools



Useful Dev Tools

- VSCode
  - Vetur
  - TSlint
  - Prettier
- ATOM
  - language-vue
- Vue Dev Tools
- StoryBook



Thank you!

jangkeehyo@gmail.com