1. 학습모듈
   1. Angular: Typescript
   2. Javascript 표준: ECMA Script 6
   3. Vuex: 상태관리를 위한 Opensource
   4. Axios: server랑 http 통신을 하기 위해서
   5. Router: 화면 전환을 위해서 필요한 기능
2. Library download 방법의 변화
   1. Npm, yarn
   2. Bundling 해주는 도구가 필요하다
      1. webpack
         1. 여러 Javascript 같은것도 .js하나로 만들어줘야하고
         2. Saas 같은 파일들도 css로 변환해줘야한다.
3. Client 와 Server 를 별도 프로젝트로 개발
   1. Client를 위한 Dev Server가 필요해졌다.
      1. webpack이 제공한다.
   2. Server Side Rendering
      1. 기존: JSP, Thymeleaf를 사용하여 그려서 내려준다.
   3. Client Side Rendering
      1. Ajax 통신: Rendering은 Client에서 하겠다.
      2. Server는 Data(json)을 전달하고, client에서 Rendering한다.
4. Compiler가 필요하다.
   1. javascript compiler가 필요하다.
      1. React, Vue.js: ECMA(에크마) Script6 사용
         1. Babel Compiler 사용, ES6 🡪 ES5 변환 (Browser의 구버전 호환을 위해..)
         2. Bable Loader: webpack이 제공
      2. Angular: Typescript(Misicrosoft 에서 만든..) 사용
         1. tsc 컴파일러 사용: ts 🡪 ES5로 변환
5. React vs Vue.js vs Angular 2+
   1. React
      1. 생태계가 넓다.
      2. HTTP Client, Router, 상태 관리 등의 기능이 내장되어 있지 않고, Core만 제공한다.
      3. 개발자가 원하는 Stack을 맘대로 골라서 사용할 수 있다.
   2. Angular 2+
      1. Type script 사용
      2. Router, HTTP Client 등 필요한 모든 기능을 framework안에 제공
   3. Vue.js
      1. Evan You 가 AngularJS의 대체재를 위해 만들었다.
      2. Core 제공하고 다른 건 공식 Library 형태로 제공한다.
      3. React와 Angular의 중간 형태로 Library를 제공하며 크기가 작다.
6. MVVM pattern
   1. Model – View – ViewModel
   2. HTML DOM이 view 역할
   3. Javascript가 Model 역할
   4. Javascript 객체와 HTML DOM간의 동기화를 vue.js가 하는데 이를 ViewModel 역할
      1. 동기화 처리시에 virtual dom 기법을 사용한다.
      2. Virtual DOM vs Real DOM
         1. <https://www.youtube.com/watch?v=BYbgopx44vo>
   5. state management를 효율적으로 하기 위해 vuex를 사용한다.
7. ECMA 6
   1. const & let
      1. const: 상수, 한번 선언하면 변경할 수 없음
         1. object 이거나 array로 사용될 경우, 값을 변경할 수 있다.
      2. let: 값이 계속 변경
         1. let vs var
            1. var

같은 이름의 변수를 여러 번 선언해도 에러 안남

scope이 function 단위 이므로 block { } 을 벗어나도 사용 가능

* + - * 1. let

같은 이름의 변수를 다시 사용하면 에러 발생

block 내에서만 사용할 수 있음.

* 1. Hoisting
     1. 코드라인 순서와 상관없이 선언한 함수와 변수를 해석기가 있는 상단에 있는것으로 인식한다.
        1. 함수 선언식

function willBeOveridden() {

return 10;}

* + - 1. 함수 표현식

var sum = function() {

return 10 + 20;}

* 1. Arrow function
     1. Example)

function sayHello(msg) {

return 'Hello ' + msg;

}

console.log(sayHello("Vuejs"));

let sayHello1 = msg => 'Hello ' + msg;

console.log(sayHello1("뷰1js"));

let sayHello2 = msg => {return 'Hello ' + msg};

console.log(sayHello2("뷰2js"));

let sayHello3 = msg => ('Hello ' + msg);

console.log(sayHello3("뷰3js"));

* 1. Array 다루기
     1. map 함수 (function stye)
        1. Example)

// map(), filter(), reduce() 함수 사용

let myArr = [10, 20, 40, 60, 70, 80, 100];

let result = myArr.map((val, idx) => (val \* idx));

console.log(result)

* + 1. filter 함수 (predicate style)
       1. 조건에 맞는 값을 필터링 해준다.
       2. Example)

let result2 = myArr.filter(val => val % 4 == 0);

console.log(result2);

* + 1. reduce 함수 (function style)
       1. 주어진 조건으로 줄인다.
       2. Example)

let result3 = myArr.reduce((prev, curr) => prev + curr);

console.log(result3);

* 1. Lamda 와 stream
     1. Spring MVC
        1. 동기 방식(Synchronous) 🡪 기존 방식
        2. 요청 하고 받는식
     2. Spring WebFlux
        1. 비동기 방식(Asynchronous) 🡪 node.js와 같은 비동기 방식
        2. publisher: 보내는 쪽은 계속 보내
        3. subscribe: 보내는 데이터를 구독신청 해서 받겠다. (받고 싶은 data size 등을 조절할 수 있다.)
        4. Mongo DB를 많이 사용한다.
        5. ReactiveX
        6. RxJava (Netflex)
  2. For Loop

let myArr = [10, 20, 40, 60, 70, 80, 100];

for(let val in myArr) {

console.log(val);

}

for(let val of myArr) {

console.log(val);

}

* 1. Default Parameter Type

let volume = (l, w = 3, h = 4) => (l \* w \* h);

console.log(volume(2));

* 1. Template literal
     1. ${} 를 써서 정적인 값과 동적인 변수를 같이 조합해서 쓸 수 있다.

let host = 'aa.com';

let port = 8080;

let database = 'mydb';

const db = 'http://' + host + ':' + port + '/' + database;

console.log(db);

//`는 백픽, ‘ 아니다.

const db2 = `http://${host}:${port}/${database}`;

console.log(db);

* 1. Destructuring Assignment
     1. 해체
     2. Example)

let a, b, rest;

[a, b, ...rest] = [10, 20, 3, 4, 5];

console.log(a);

console.log(b);

console.log(rest);

// packing

let foo = ["one", "two", "three"];

// unpacking

let [foo1, foo2, foo3] = foo;

console.log(foo1);

console.log(foo2);

console.log(foo3);

* 1. 값 교환 (Swap)
     1. destructuring assignment - swapping

let a = 1;

let b = 3;

[a, b] = [b, a];

console.log(a);

console.log(b);

* 1. 객체 분해
     1. key 값에 맞춰서 분해

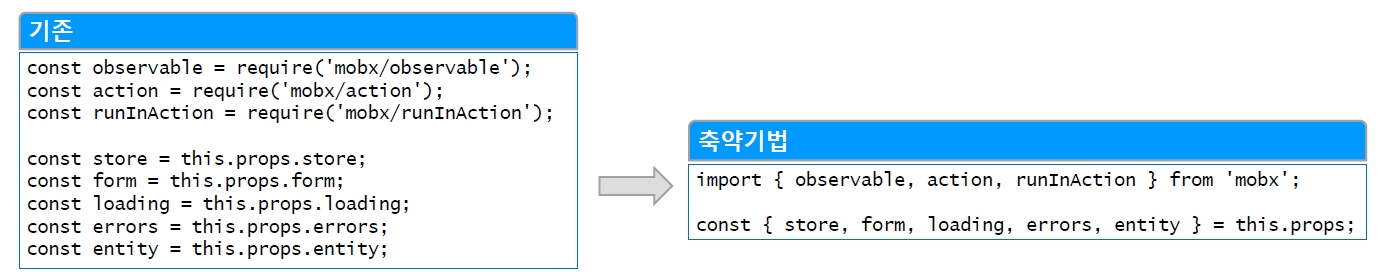
let obj = {p: 42, q: true};

let {p, q} = obj;

console.log(p);

console.log(q);

* 1. 축약 기법



* 1. 전개 연산자(spread operator)
     1. … 를 이용하여 배열의 값을 대체할 수 있다..

// joining arrays

const odd = [1, 3, 5];

const nums = [2 , ...odd, 4 , 6].concat(odd);

console.log(nums);

* + 1. 복사

// cloning arrays

const arr = [1, 2, 3, 4];

const arr2 = arr.slice();

const arr3 = [...arr];

console.log(arr2)

console.log(arr3)

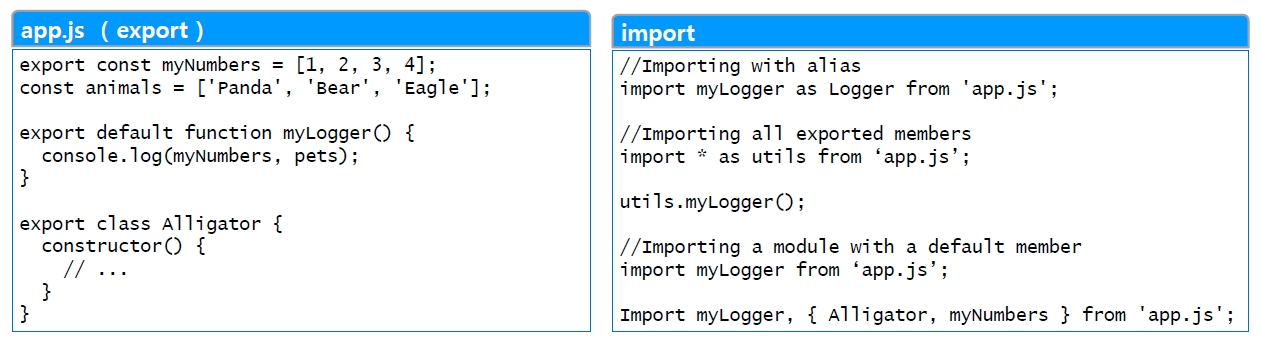
* + 1. 원본 자르기
       1. slice()는 원본은 잘리지 않지만 splice()는 원본도 잘린다.

let myArr2 = [10, 20, 30, 40, 50];

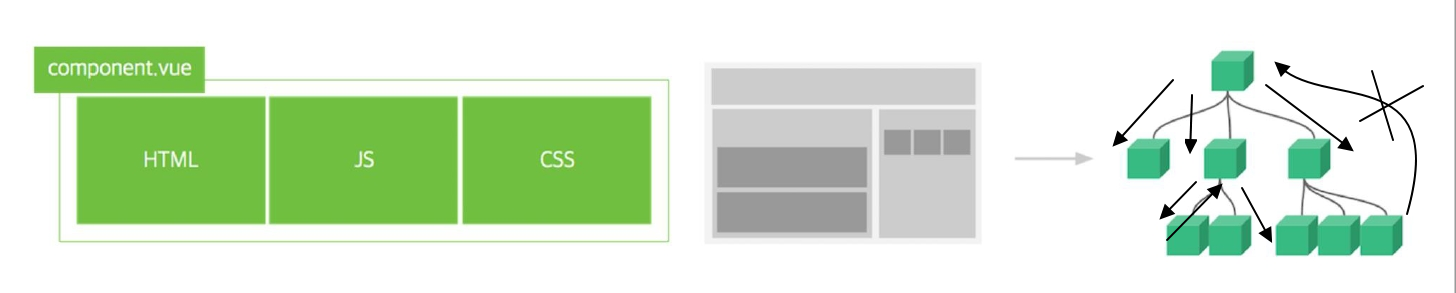
console.log(myArr2.splice(2));

console.log(myArr2);

* 1. import / export
     1. 각 각의 javascript 파일들을 모듈화 하여, export 한 모듈을 import를 통해 불러와 쓸 수 있게 한다.
     2. as 를 사용하여 alias를 부여할 수 있다.
     3. \* 를 사용하여 모두를 import 할 수 있다.
     4. default 를 붙여서 export 할 때는 { } 가 제외되지만 default를 안하면 { } 를 붙여서 import 한다.



1. computed vs methods vs watch
   1. computed property는 dependent 값이 바뀌지 않는 한 호출되지 않는다.
   2. 반복되서 값을 가져오는데 시간이 오래걸리는 것은 computed property를 사용하는 것이 좋다.
2. Component
   1. Vue.JS는 단일 파일 Component를 추천한다.
   2. 확장자가 vue인 파일로 만들어서 HTML, JS, CSS 코드로 구성한다.
   3. CSS는 상위 CSS 영향을 받지 않는다.
   4. Data 전달은 바로 위아래 컴포넌트들 끼리만 가능하다.



1. v-model
   1. 상태변수(state variable)를 선언하고, 그 변수와 html dom을 연결한다.
   2. 역할론
      1. 변수 선언 (개발자)
      2. 변수와 HTML DOM 연결(개발자) – directive를 사용
      3. 변수와 HTML DOM 간의 동기화(vue.js) – ViewModel
2. Vue CLI
   1. package.json 파일 생성
3. 시작하기
   1. 01\_FirstProject
      1. cmd: vue create vue-hello-app
      2. package.json
         1. d

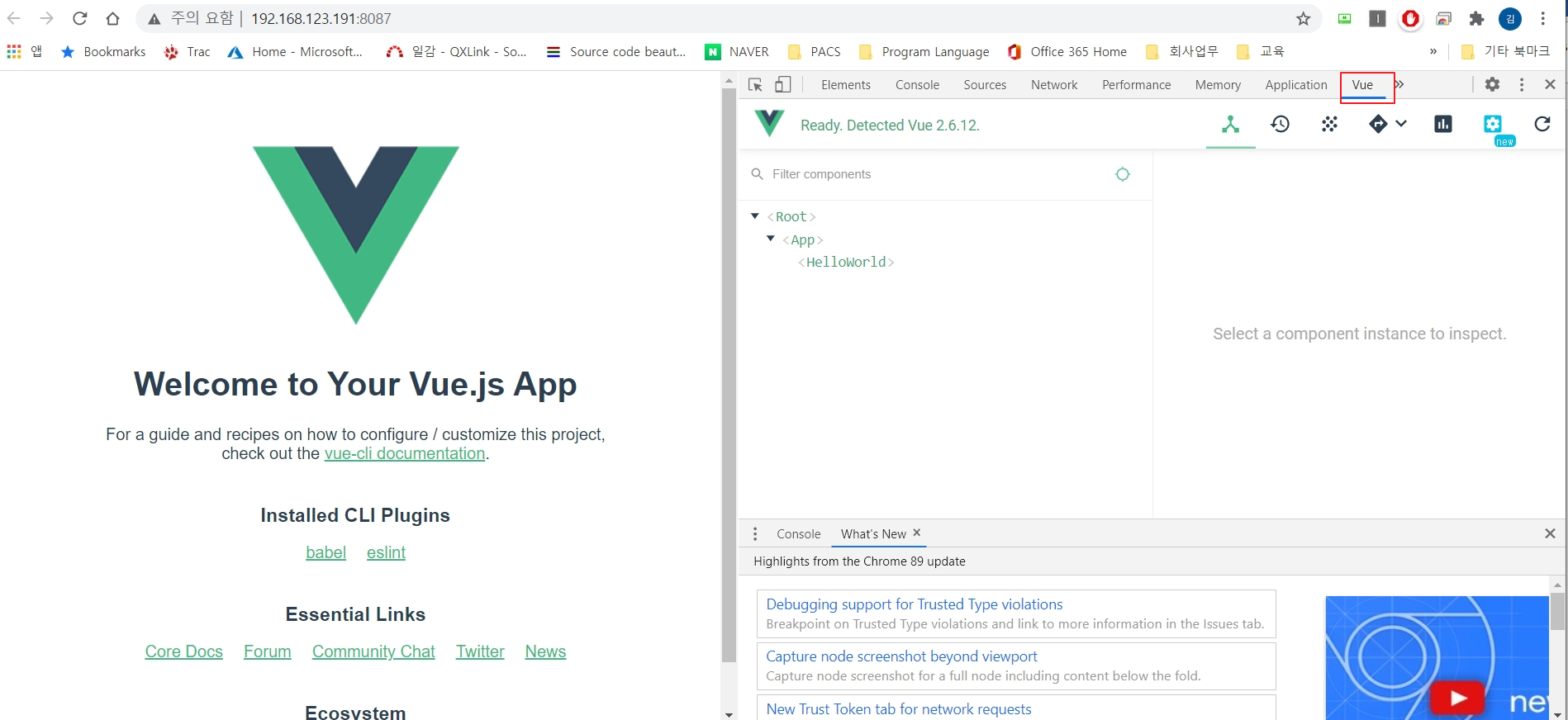
  "scripts": {

    "serve": "vue-cli-service serve",

    "build": "vue-cli-service build",

    "lint": "vue-cli-service lint"

  },

* + 1. 개발Dev server start하고, compile
       1. npm run serve
          1. port number를 바꿀 때, npm run serve -- --port 8087
       2. npm run build
          1. 운영에 배포(Deploy)하기 위한 작업
       3. 
       4. http://localhost:80817 요청하면
          1. public/index.html 요청 (Single page application)이라 html은 1개

<div id="app"></div>

* + - * 1. src/main.js

index.html과 App.vue를 연결해주는 역할

webpack에 main.js가 제일 먼저 실행되도록 설정되어 있다.

* + - * 1. src/App.vue

Root component

* + - * 1. src/components/HelloWorld.vue

Vuejs Component

* + - * 1. workflow

index.html –(main.js)🡪 App.vue 🡪 HelloWorld.vue

* + - * 1. Data 교환 규칙

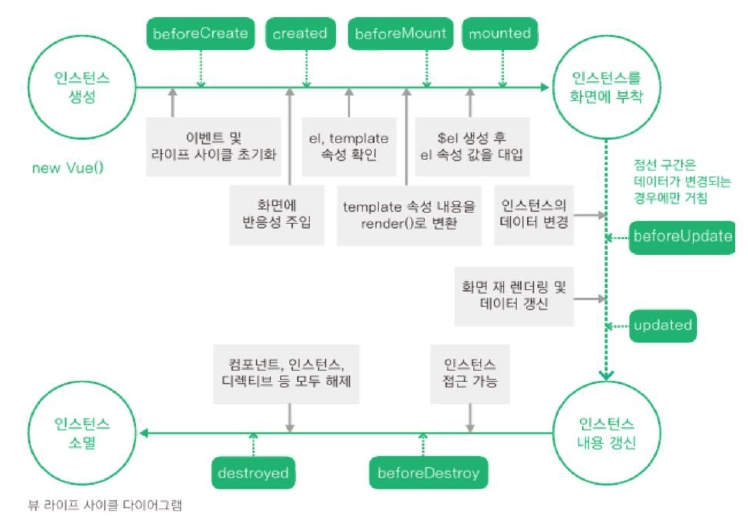
부모 Component가 자식 component에게 props를 전달할 때,

부모 🡪 자식 msg-title (snake style)

자식이 가져다 쓸 때,

자식 🡪 부모 msgTitle (camel style)

1. Life-Cycle



1. http test
   1. Postman 사용
      1. 조회
         1. method: GET
         2. http://localhost:4500/api/todos
         3. http://localhost:4500/api/todos/2
   2. 등록
      1. method: POST
      2. http://localhost:4500/api/todos
      3. header
         1. content-type application/json
      4. 2)body - raw

{

"item": "Todo4",

"completed": true

}

* + 1. 수정
       1. method: PUT
       2. http://localhost:4500/api/todos/1
       3. header
          1. content-type application/json
       4. body - raw

{

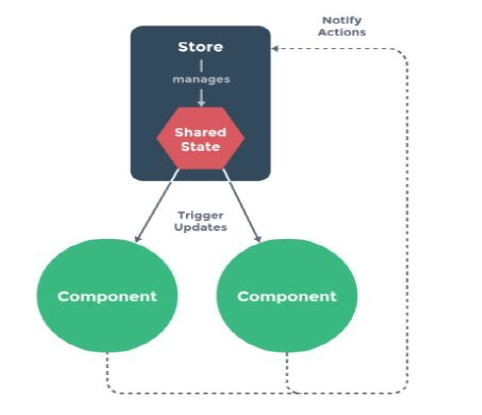
"item": "Todo11",

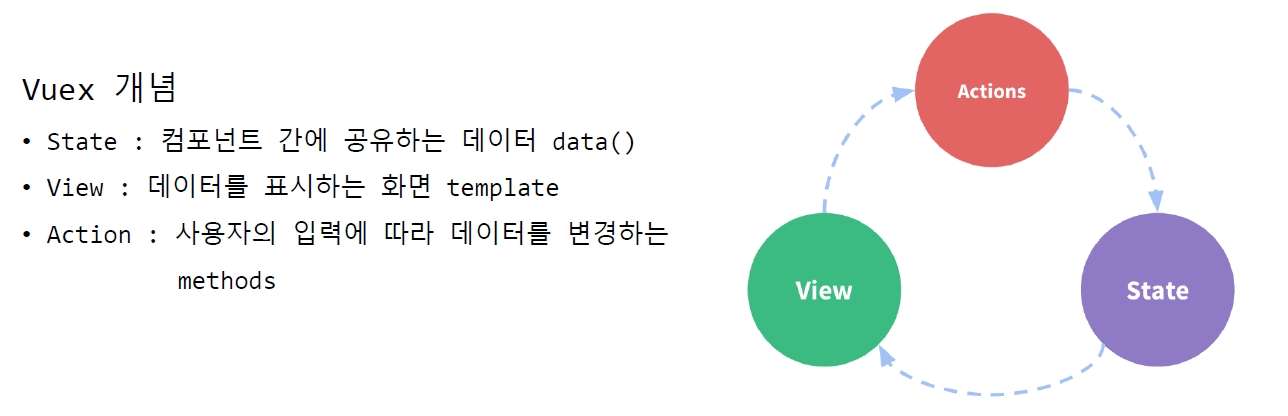
"completed": true

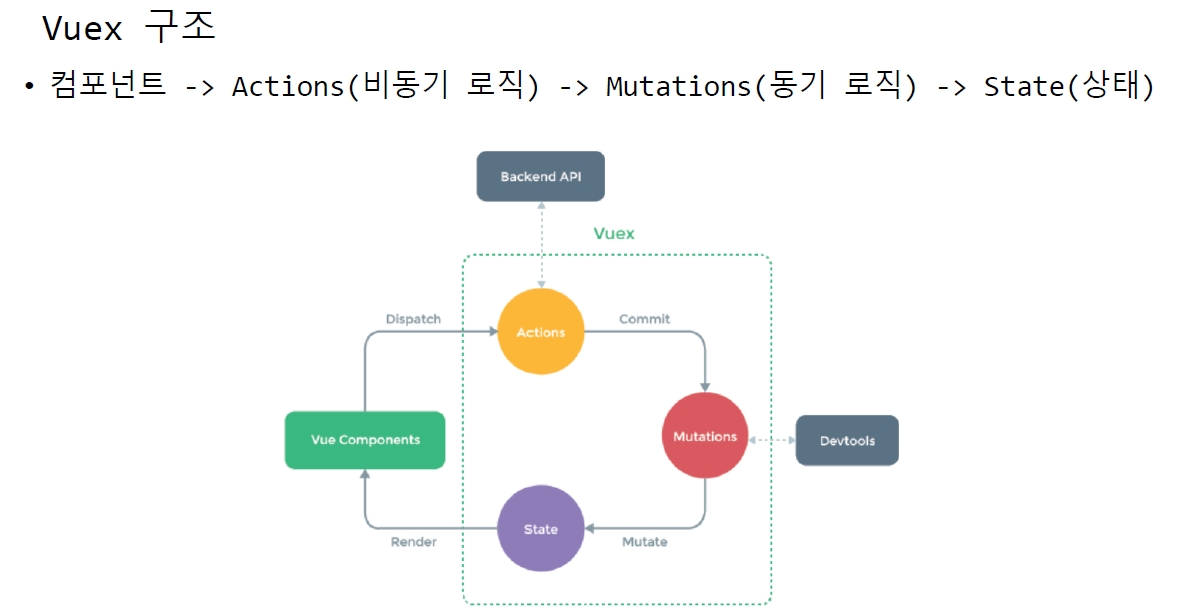
}

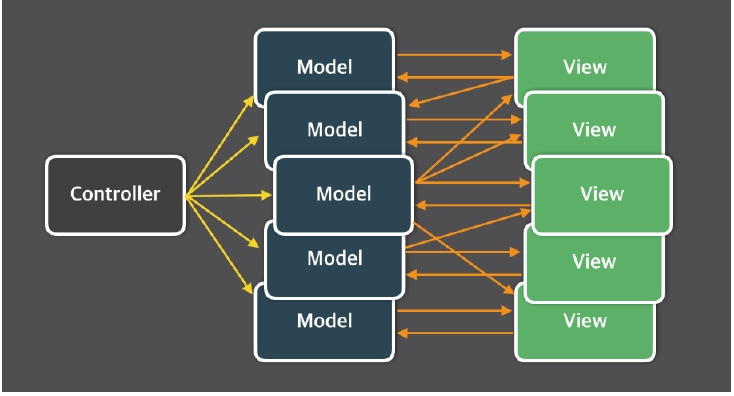
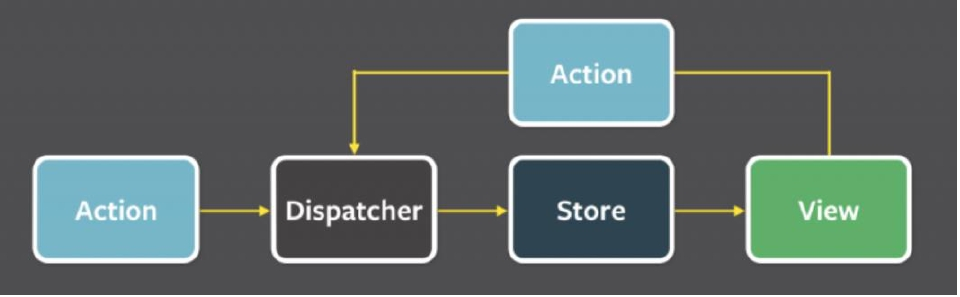
* + 1. 삭제
       1. method: DELETE
       2. 특정 ID 삭제
          1. http://localhost:4500/api/todos/4
       3. 전체삭제
          1. http://localhost:4500/api/todos

1. VUEX
   1. 개요
      1. Vue.js Application에 대한 상태 관리 패턴 + 라이브러리



* + 1. 부모 자식 간에만 Data를 전달 할 수 있으므로 제3의 저장소에 저장해서 상태관리를 하자.  
       
  1. 구조
     1. state: 변수
     2. Action
        1. async method
        2. commit을 통해 mutations의 setter method를 호출
        3. Server와 http 통신하는 method 구성
     3. Backend API: Server
     4. Mutations:
        1. Setter Method
        2. Sync Method
        3. 상태 변수를 변경
     5. State: Vue component에 변경 알림



* 1. MVC 패턴 문제점  
     
     1. 양방향 데이터 흐름의 문제
        1. 복잡하지 않으면 문제가 안되나, 복잡해지면 데이터 변경 이력의 추적이 힘들다.
  2. 위 문제를 개선 🡪 FLUX  
     
     1. 데이터 흐름을 단방향만..
     2. VUEX, Redux
     3. 역할
        1. View: Event 발생
        2. Action: 특정 Command 요청
        3. Dispatcher: 전달
        4. Store: state를 관리.
  3. Helper사용법
     1. method를 좀더 간결하게 사용할 수 있게 하는
     2. import { mapState } from 'vuex'
     3. import { mapGetters } from 'vuex'
     4. import { mapMutations } from 'vuex'
     5. import { mapActions } from 'vuex'

1. 개발(Development)과 운영(Production)을 위한 설정파일 작성
   1. 경로
      1. package.json 파일이 있는 위치와 동일
   2. 개발
      1. .env.development
         1. VUE\_APP\_APIURL=http://localhost:4500
      2. todo.js
         1. const api\_url = process.env.VUE\_APP\_APIURL
   3. 운영
      1. .env.production
      2. > npm run build
         1. 묶은 파일이 dist라는 폴더에 생김
2. Short-cut
   1. <vue> component 기본 Template 생성 snippet
   2. Ctrl + D 전체 변경
3. Router 추가
   1. router module 설치
      1. npm install –save vue-router
   2. router 관련 코드 추가
      1. vue add router
      2. main.js에서 router를 render 앞으로 위치 변경
      3. 주의 App.vue 기존것이 지워진다.
   3. index.html –(main.js: router/index.js, / 🡪 Home(view의 Home component) 🡪 HelloWorld 🡪 TodoHeader, TodoList 등등 ) 🡪 App.vue
   4. Router link
      1. App.vue

<router-link to="/">Home</router-link>

<router-link to="/about">About</router-link>

1. node-module
   1. axios, vue-axios
      1. http client library
   2. express
      1. node 기반의 web framework
      2. express object를 이용하여 server side를 구성할 수 있다.
   3. cors
      1. Cross-Origin Resource Sharing
      2. Single Origin Policy를 우회하기 위한 기법
      3. Origin
         1. URI 스키마(http) + hostname (localhost) + Port (8080, 8081)
      4. 개발 모드에서 CORS 적용
         1. express, cors 의존성 설치
         2. Example)

const cors = require('cors');

const app = express();

app.use(cors());

* + - 1. Sprint에서 쓰는 방법이 따로 있음.

1. 주요 command
   1. client
      1. npm run serve
      2. npm run build
   2. server
      1. node todo.js