

# <Term Project: Survey on Programming Paradigms>

웹 프론트엔드의 모던 프레임워크 비교

(React,Angular,Vue를 이용한 프론트엔드 작성의 장단점 비교)

-개발 프레임워크 관련-

과목명: 프로그래밍언어(다)

소속: IT대학 컴퓨터학부

제출일: 2019.05.11

학번: 20142303

이름: 권태형

# 웹 프론트엔드의 모던 프레임 워크 비교

[ React, Angular, Vue를 이용한  
프론트엔드 작성의 장단점 비교 ]

과목명: 프로그래밍언어(다)  
소속: IT대학 컴퓨터학부  
제출일: 2019.05.11  
학번: 20142303  
이름: 권태형

# 목차

1. React/Angular/Vue란 무엇인가?
  - 1-1. 프레임워크의 필요성
  - 1-2. React/Angular/Vue의 개요
  - 1-3. React/Angular/Vue의 구현방식
2. React/Angular/Vue비교
  - 2-1. 트렌드 비교
  - 2-2. 학습 코스트 비교
  - 2-3. 성능 비교
3. 결론 및 추가정보



-chapter-

# React/Angular/Vue란 무엇인가?

기본적인 웹 프레임워크 비교 전에 알아야 할 지식들



# 프레임워크의 필요성

웹 프론트엔드에 프레임워크가 필요한 이유



# 프론트엔드와 프레임워크

- 프론트엔드와 프레임워크란?
- 프론트엔드(Front-end)
  - 서버에서 데이터를 가져와서 사용자한테 보여주거나 사용자한테 받은 입력을 받아서 서버에 보내주는 **사용자 인터페이스** 부분을 말함 <2>
  - 웹을 예시로 들자면 홈페이지에 들어갔을 때 눈에 보이는 부분이 바로 프론트엔드 부분



# 프론트엔드와 프레임워크

- 프레임워크

- 문제 해결을 위한 클래스와 그에 따른 라이브러리가 합쳐진 것 <1>
- 프레임워크 자체만으로는 프로그램이 돌아가지 않으며  
프레임워크가 정의한 규칙에 따라 기능을 추가해야 한다. <2>
- 새로운 언어가 아니라 기존에 사용하는 언어를 기반으로 만들어짐  
(Python의 Django, Java의 Spring, .Net 프레임워크등)
- 특정 목적에 맞는 프레임워크는 개발 속도를 빠르게 해준다



# 프레임워크

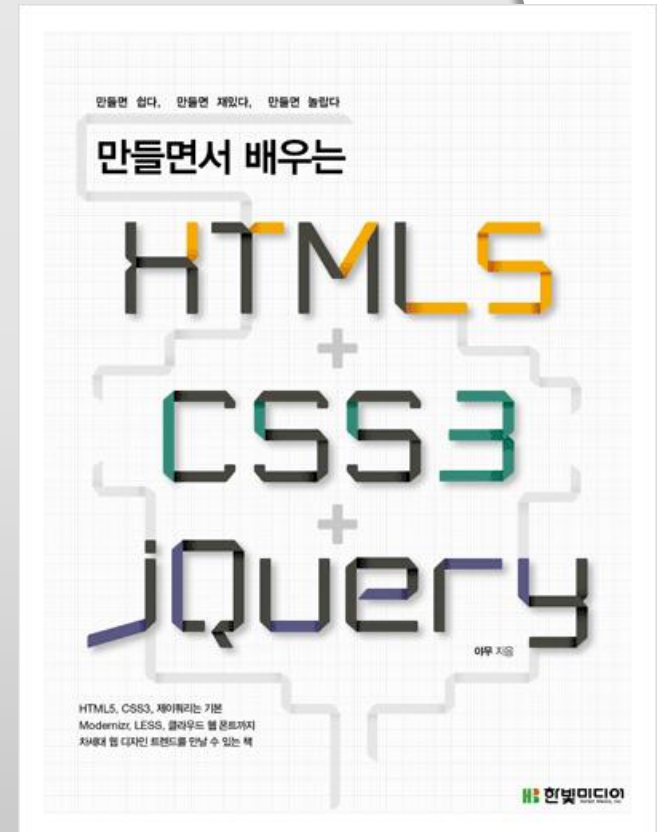






# 웹 개발에 프레임워크가 필요한 이유?

- HTML, CSS, Javascript로만 안되는 이유
- 프레임워크가 정말 개발을 편하게 만들어 주는가
- 프레임워크 쓰면 오히려 무거워져서 느려지지 않는가
- 애초에 자바스크립트를 쓰는 이유





# 첫번째 이유, 효율성

- 기존에 사용되던 jQuery의 한계 <4>
  - 처음에 다양한 기능들을 내세우면서 많은 인기를 얻음.
  - jQuery의 첫번째 장점: DOM접근의 간편함  
그러나 이런 기능도 슬슬 많은 프레임워크에서 지원해주기 시작
  - jQuery의 두번째 장점: 브라우저간 호환성  
이것도 많은 브라우저가 표준을 지키기 시작하면서 자연스레 해결되기 시작
  - jQuery의 단점이 드러나기 시작
  - 느리고 코드 유지비용도 많이 든다



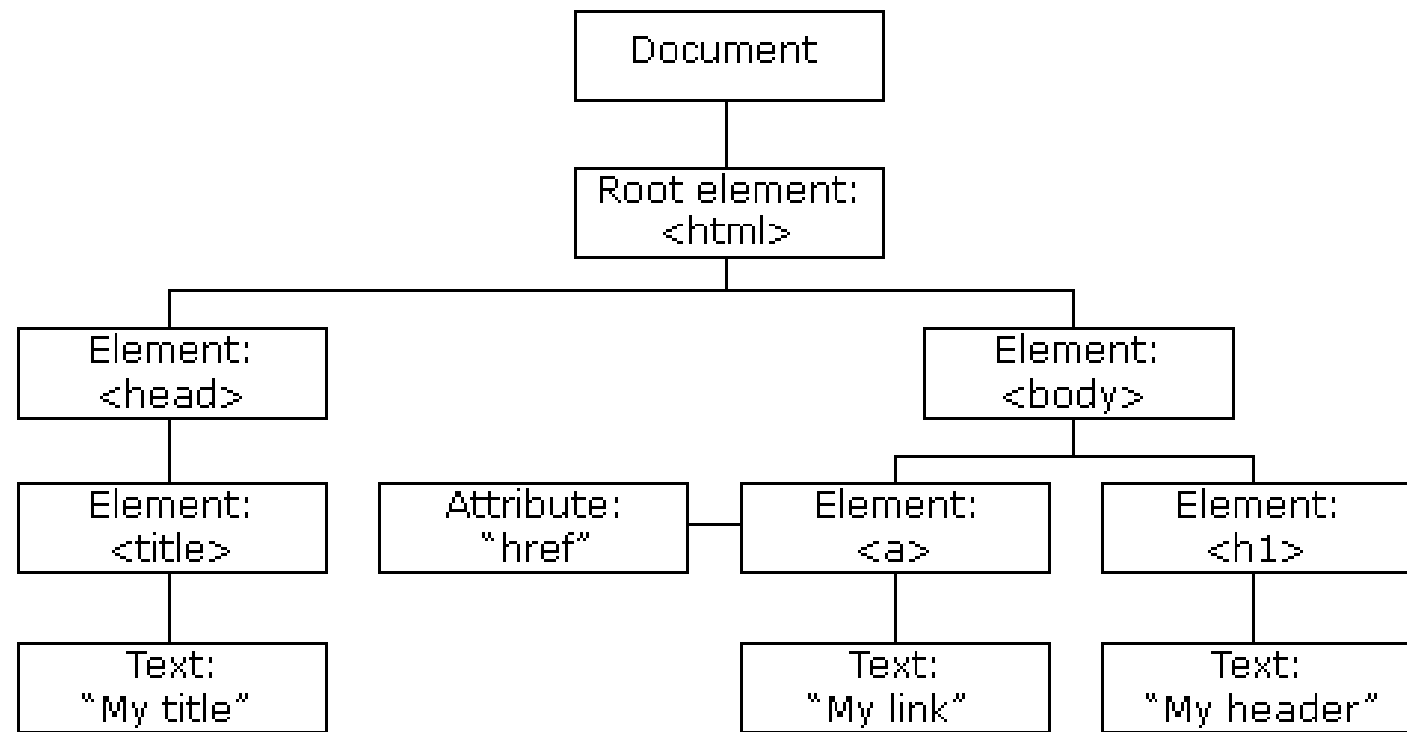
# DOM

<5>

- 넓은 의미로는 브라우저가 html을 인식하는 방향
- 좁은 의미로는 문서에서의 객체의 집합을 뜻한다.
- 웹 관점에서의 문서 객체란? <body>, <html>같은 태그들을 JS가 인식할 수 있는 객체로 만들어 진 것.
- DOM탐색법은 일반적으로 매우 비효율적이고 느리다.



# DOM Tree



```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href = "zelda.or.link">
      My link
    </a>
    <h1>My header</h1>
  </body>
</html>
```

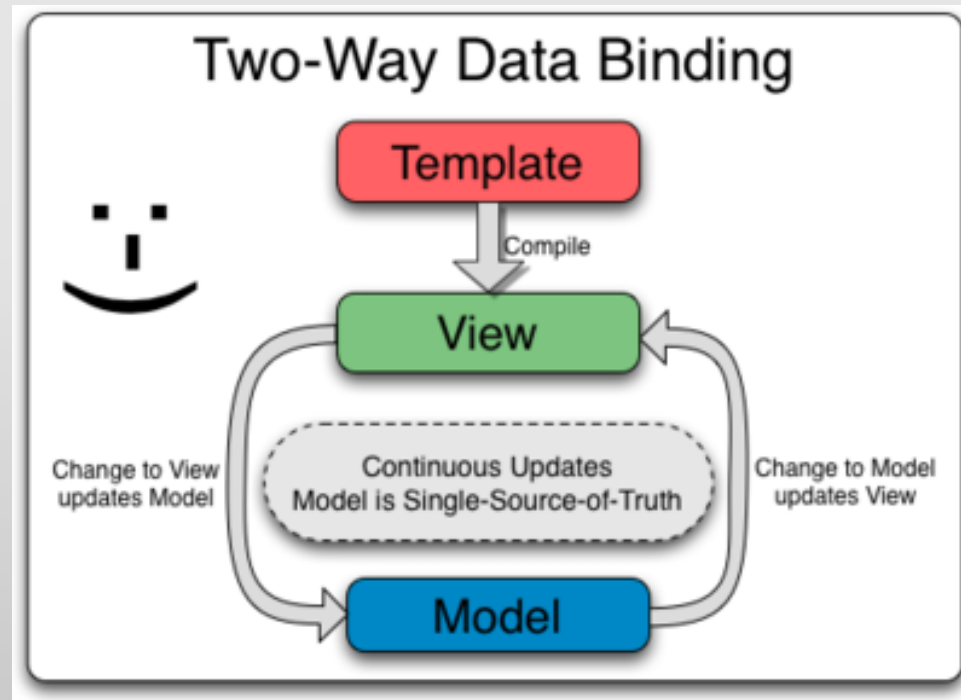
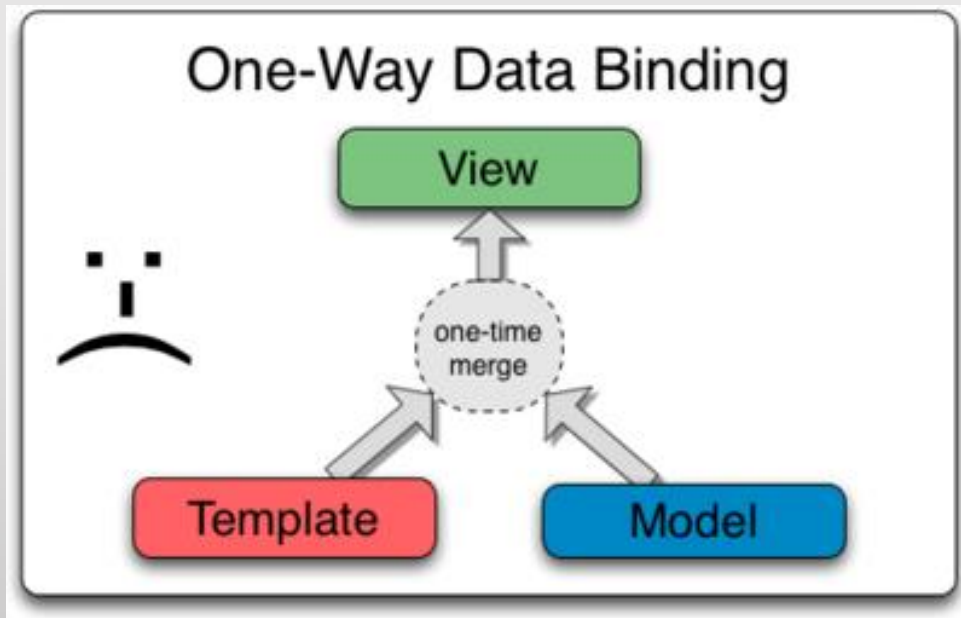


# 첫번째 이유, 효율성

- React의 해결방식 <6, 8>
  - DOM의 해석과 렌더링은 브라우저 입장에서 꽤 큰 비용을 요구한다.
  - Virtual DOM은 이것을 최적화 해준다.  
예를 들어 사용자가 DOM의 내용을 바꾸면 이를 기존의 DOM은 하위 요소들까지 전부 다시 렌더링 하지만,  
Virtual DOM은 바뀐 부분만 추려내서 렌더링하게 도와준다.
- AngularJS의 해결방식 <3, 7>
  - AngularJS는 jQuery lite를 포함함으로써 느린 속도를 일부 개선하였습니다.
  - DOM의 접근에 대한 초점보다 데이터의 변화에 초점을 두었습니다.
  - Two Way Binding으로 데이터 변경만으로 뷰를 바꿀 수 있음



# Two Way Binding



<36>

<https://dalkomit.tistory.com/156>

<3>



# 두번째 이유, 편의성

<0>

- 최근 웹에서 요구하는 것
  - 동적 렌더링
  - 좋은 재사용성
  - 상태 변화도 감지
  - 타입 체크
  - 테스트 자동화
- 각각의 기능을 하는 도구들을 다 알아야 한다



## 두번째 이유, 편의성 <0>



프레임워크는 이 많은 것들을 쉽고 간결하게 만들어 준다





# 세번째 왜 JS를 사용하는가

<12>

- 구글이 개발한 **V8** 엔진개발(C++기반)
  - 웹브라우저 내에서 Javascript를 더 빠르게 수행하기 위해서 개발
  - 초기의 Interpreter해석방식의 **Javascript**를 **JIT방식으로 구동** 시키는 엔진
  - 현재 JS표준을 제일 잘 가동시킬 수 있는 엔진
- V8을 이용한 **Node.JS**의 개발
  - **Javascript**를 런타임으로 돌릴 수 있게 되었다
  - 기존 서버사이드 플랫폼에 비해 **가볍고 효율적**이다.



# 정리

- 2000년대 초반에는 **VanillaJS**가 널리 쓰이고 있었다.  
VanillaJS == Pure JS라고 보긴 어렵지만 **거의 순수한 JS**를 쓰고 있었던 셈.
- 2006년에 **jQuery**라이브러리가 등장, 기존 **JS를 더 쉽게** 만들어주고, **DOM탐색 지원, 멀티 브라우저 지원**(어디에서나 똑같이 돌아가는 기능)<11>
- 2009년에 **AngularJS 프레임워크**의 등장  
jQuery만 사용했을 때보다 **빠른 속도 보장**  
백엔드와 프론트엔드의 **역할을 확실히 구분**해줌.<7>



# 정리

- 2013년에 Node.js출시 서버사이드에 JS가 쓸만해지기 시작
- 동시에 React의 등장 Virtual DOM으로 DOM의 새로운 패러다임을 제시해줌.
- 2014년 Vue의 등장 쉽게 배울 수 있고, 깨끗한 모듈화 지원, 재사용성도 매우 좋은 장점을 들고 나옴<9>
- 2016년에 Angular(Angular2)등장 기존 AngularJS의 단점을 보완해 React와 견줄만큼의 스펙을 탑재하고 나옴<10>



# React Vue Angular의 개요

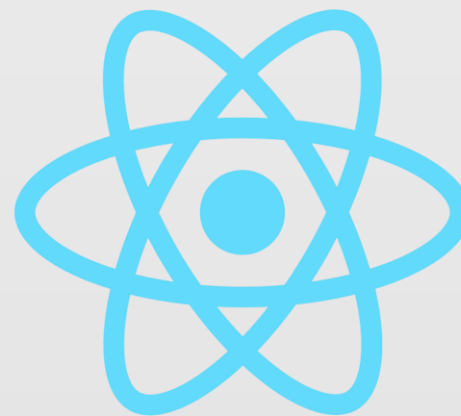
각자의 강점 알아보기



# React

<0, 13 - 1>

- 출시일: 2013년 3월
- 개발사: Facebook/Instagram
- 최신버전: 16.8.6 (March 27, 2019)
- 언어: Javascript/JSX
- 모델방식: Virtual DOM/View Engine
- 컴파일방식: JIT/TreeShaking Webpack



<37>



# React만의 장점

<6>

- 사실 React는 프레임워크 보다는 라이브러리에 가깝다
- Virtual DOM의 지원으로 빠른 속도를 보장
- JSX의 사용(PHP의 기반)  
일반 자바스크립트에 리턴을 HTML코드로 할 수 있게 해준다  
이를 통해 결과물을 직관적으로 알 수 있고 코드의 가독성을 크게 올려줌



# Angular(Angular2)

<0, 13 - 2>

- 출시일: 2016년 6월
- 개발사: Google
- 최신버전: 8.0.0-rc.2 (2019-04-29)
- 언어: Typescript/Javascript
- 모델방식: MVVM(two way binding)/Change Detection
- 컴파일방식: JIT/AOT/TreeShaking with ng cli



<38>



# Angular만의 장점

<7, 10>

- 양방향 데이터 바인딩을 하면 **Cascade효과(도미노)**를 일으킬수 있음으로 단방향을 지원하기 시작
- **서버 렌더링**의 지원으로 리액트와 비슷한 스펙을 가지기 시작
- **템플릿과 코드의 분리**로 개별 관리의 가능
- 마크업 언어는 **HTML표준**을 사용한다
- 양날의 검 Typescript를 사용한다  
(JS말고 또 배워야함 VS Type Checking을 지원)



# Vue

<0, 13 - 3>

- 출시일: 2014년 2월
- 개발자: Evan you
- 최신버전: v1.6.0(2019.03.29)/ v2.0.0-rc.1 (2019.04.24)
- 언어: Javascript/JSX호환
- 모델방식: MVVM/Virtual DOM/Vuex
- 컴파일방식: JIT/TreeShaking Webpack



<39>



# Vue만의 장점

<9, 14>

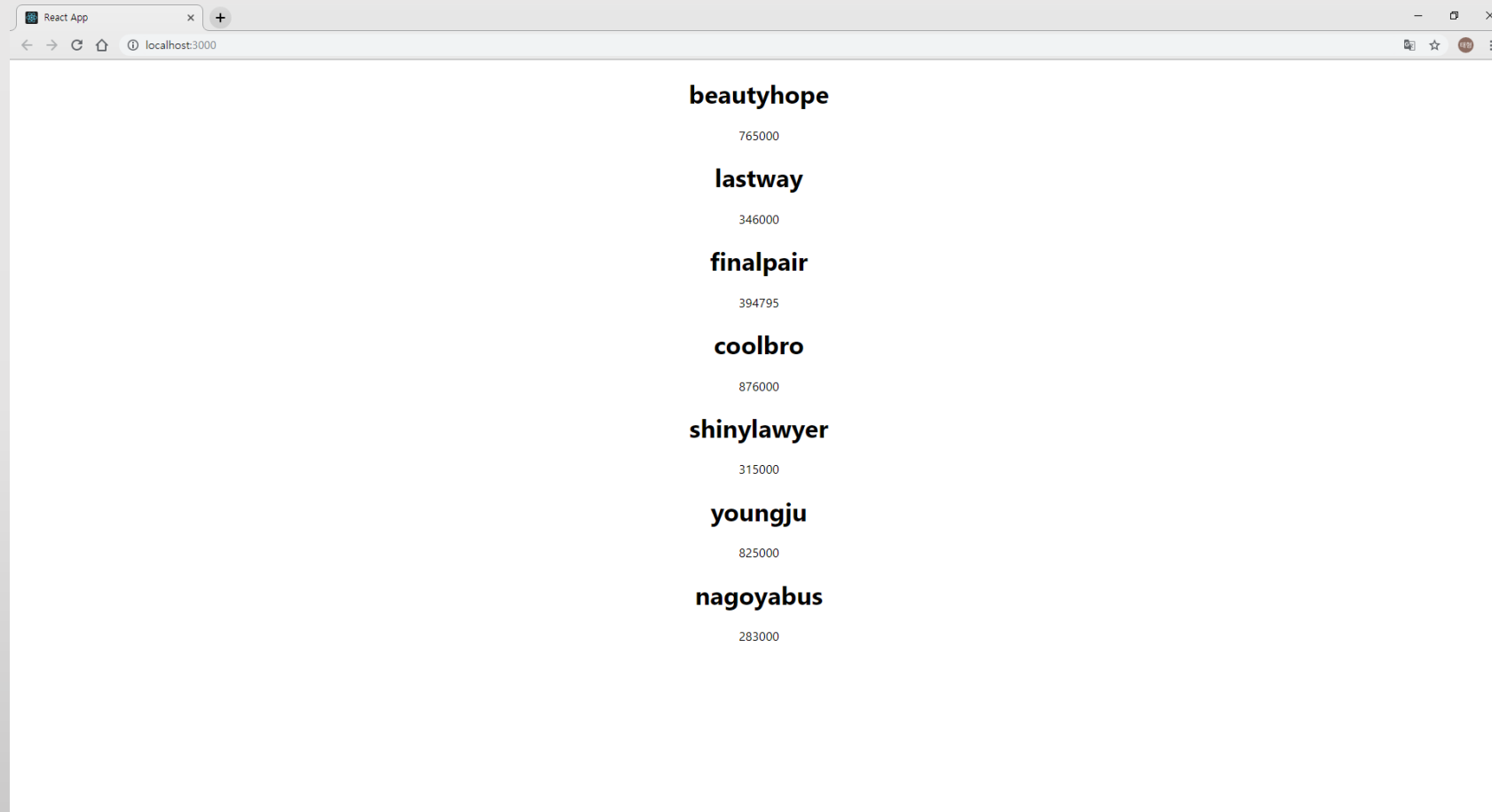
- 학습 곡선이 매우 낮다. 즉 언어 적응기간이 매우 짧음
- 하나의 파일로 컴포넌트를 관리할 수 있다.  
마크업/스타일/스크립트파일 한 개 == 하나의 컴포넌트
- React의 Virtual DOM, Angular의 MVVM 동시 지원  
장점만 가져와서 탄생한 프레임워크



# React Vue Angular의 구현 방식

코드로 구현방식 알아보기

# 구현 목표



A screenshot of a web browser window displaying a list of users and their scores. The browser's address bar shows 'localhost:3000'. The page content is a simple list with two columns: user names and scores.

<b>beautyhope</b>	765000
<b>lastway</b>	346000
<b>finalpair</b>	394795
<b>coolbro</b>	876000
<b>shinylawyer</b>	315000
<b>youngju</b>	825000
<b>nagoyabus</b>	283000

웹서버에서 데이터를 읽어와서 유저이름과 획득 점수를 출력하는 목표



# 구현 목표

<16>

- Express로 서버를 구현(Node js상)
- npm대신에 yarn을 패키지 매니저로 이용
- node init
- yarn add express
- yarn express app\_name으로 프로젝트 생성
- package.json에 코드를 추가해줌
  - “scripts”: {
    - “start”: “node index.js” // yarn start 하면 이 프로젝트의 index.js가 실행됨
- 마지막으로 index.js에 코딩



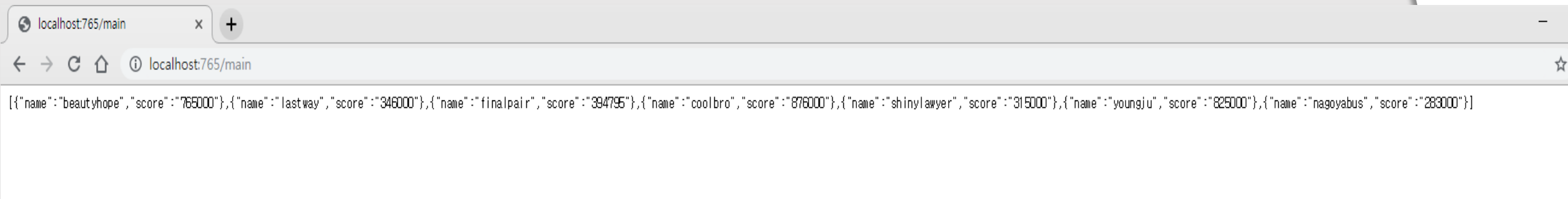
# 구현 목표

```
const express = require("express");//Express를 가져옴
var app = express();//Express 객체를 생성
App.use( " /main " ,(req,res,next)=>{//해당 객체에 /main요청이 들어온다면
res.header("Access-Control-Allow-Origin", "*"); //Cross Domain때문에 붙인 문장들
Res.header( " Access-Control-Allow-Headers " , " X-Requested-With " );//보안문제가 생길수 있음
res.json([{//json응답을 해줍니다
  "name":"beautyhope",//json 정의
  " score " : " 765000 "
},{...
  " name " : " lastway " ,
  " score " : " 346000 "
},{
}]
}))

app.listen(765); //이객체는 765번 포트를 사용하게 한다.
```



# http://localhost:765/main

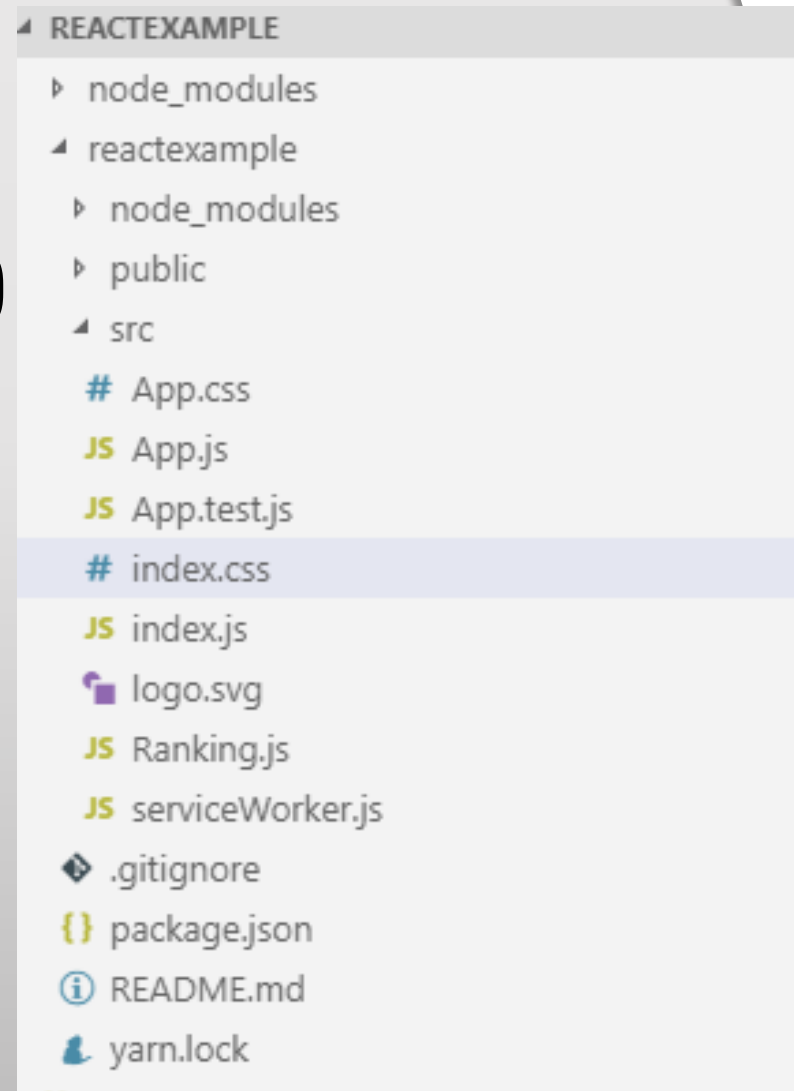
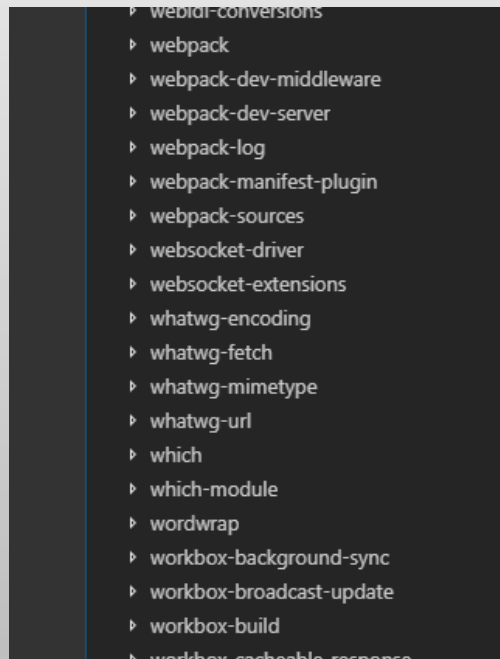
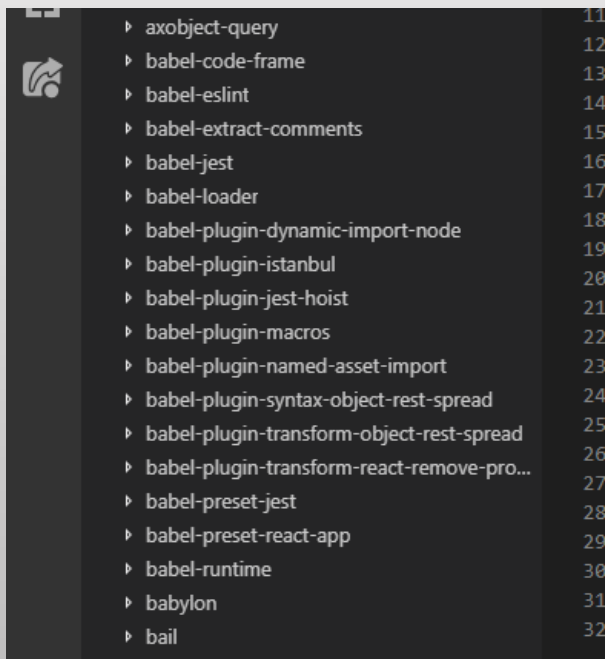


```
[{"name": "beautyhope", "score": "765000"}, {"name": "lastway", "score": "346000"}, {"name": "finalpair", "score": "394795"}, {"name": "coolbro", "score": "876000"}, {"name": "shinylawyer", "score": "315000"}, {"name": "youngju", "score": "825000"}, {"name": "nagoyabus", "score": "283000"}]
```



# React 예시 코드 생성

- node init
- yarn add create-react-app  
(Babel, Webpack을 자동으로 받아줍니다)
- yarn create-react-app app-name







# React 예시 코드 생성

- **ReactDOM**으로 Virtual DOM을 생성합니다.

App.js

JS Ranking.js

# App.css

JS index.js

×

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import App from './App';
5  import * as serviceWorker from './serviceWorker';
6
7  ReactDOM.render(<App />, document.getElementById('root'));
8
9  // If you want your app to work offline and load faster, you can
10 // unregister() to register() below. Note this comes with some
11 // Learn more about service workers: https://bit.ly/CRA-PWA
12 serviceWorker.unregister();
```



# React 예시 코드 생성

```
JS App.js  ✕  JS Ranking.js  # Ap
1  import React from 'react';
2  import Ranking from './Rankin
3  import './App.css';
4
5  function App() {
6    return (
7      <div className="App">
8        <Ranking/>
9      </div>
10   );
11 }
12
13 export default App;
14
```

- JSX문법입니다의 적용 예시
- 이 클래스가 무엇을 생성할지 예측이 바로 가능하다



# React 예시 코드 생성

```
App.js  JS index.js  JS Ranking.js x  # App.css

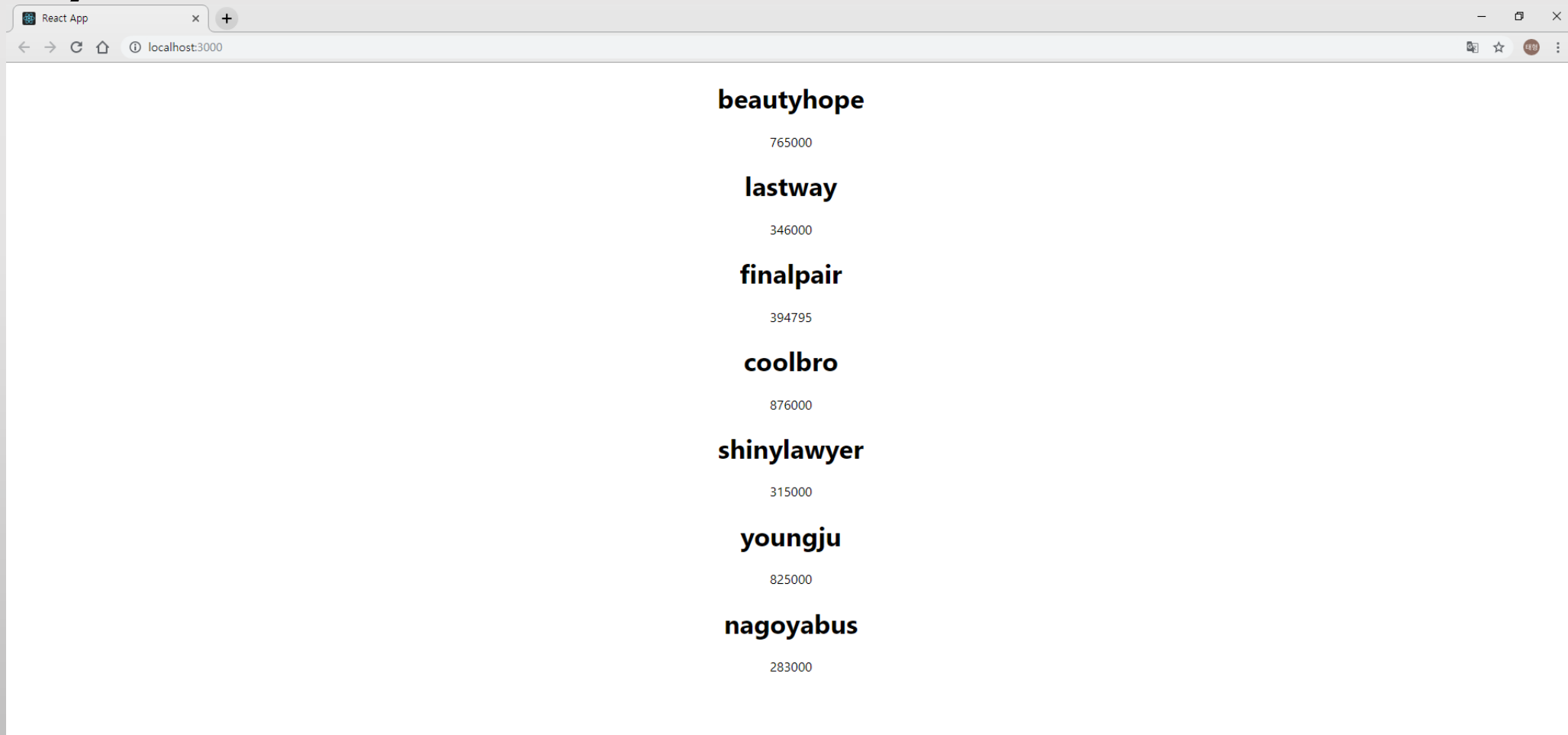
1  import React,{ Component} from 'react';
2
3  class Ranking extends Component{
4      constructor(props, context) {
5          super(props, context);
6          this.state = {board: []};
7
8          fetch("http://localhost:765/main")
9              .then(res => res.json())
10             .then(json => this.setState({ board: json }));
11     }
12     render(){
13         return (<div>{this.state.board.map((ranker,index)=>{
14             return(<div key = {index}><h1>{ranker.name}</h1>
15             <p>{ranker.score}</p></div>)
16         }}}</div>)
17     }
18 }
19
20 export default Ranking;
```

- React Component 생성
- prop은 부모가 자식에게 넘겨주는 값, 변경할 수 없다<18>
- 클래스 내의 가변 요소는 state로 들어간다. Constructor 때 미리 타입을 정해주는 걸 추천<18>
- State의 변경은 반드시 setState를 거쳐야 함
- setState, render 둘다 컴포넌트의 변화를 준다.  
(Virtual Dom 업데이트)



# React 예시 코드 생성 (localhost:3000)

## yarn start





# Angular 예시 코드 생성

<17 - 2>

- node init
- npm install -g @angular/cli(yarn 대신 npm사용)
- Yarn(npm) ng new my-app
- src/app에서부터 시작
- Typescript를 사용하는 모습을 볼 수 있다
- 간단하게 Ranker.ts 클래스 생성

```
TS ranker.ts ✕  <> app.component.ts
1  export class Ranker {
2      name: string;
3      score: number;
4  }
```

## ANGULAREXAM

- e2e
- node\_modules
- ▾ src
  - ▾ app
    - ▾ rankers
      - # rankers.component.css
      - TS rankers.component.spec.ts
      - TS rankers.component.ts
    - TS app-routing.module.ts
    - # app.component.css
    - <> app.component.html
    - TS app.component.spec.ts
    - TS app.component.ts
    - TS app.module.ts
    - TS ranker.ts
  - assets
  - environments
  - ≡ browserslist
  - ★ favicon.ico
  - <> index.html
  - 📄 karma.conf.js



# Angular 예시 코드 생성

TS app.module.ts x

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { FormsModule } from '@angular/forms';
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { RankersComponent } from './rankers/rankers.component';
7  import { HttpClientModule } from '@angular/common/http';
8
9  @NgModule({
10   declarations: [
11     AppComponent
12   ],
13   imports: [
14     BrowserModule,
15     FormsModule,
16     AppRoutingModule,
17     HttpClientModule
18   ],
19   providers: [],
20   bootstrap: [AppComponent]
21 })
22 export class AppModule { }
```

- 모듈의 추가는 app.module.ts에서 진행된다.
- Imports에도 HttpClientModule을 넣어준다.
- @는 this.을 의미한다.  
즉 this.NgModule()인셈



# Angular 예시 코드 생성

TS rankers.component.ts ✕

```
1
2 import {Ranker} from '../ranker';
3 import { HttpClient } from '@angular/common/http';
4 import { Observable } from 'rxjs';
5 import { Injectable } from '@angular/core';
6
7 //의존성 여부 설정
8 //단순히 함수 역할만 하는 클래스면 선언 안해도 되지만
9 //외부 모듈을 사용할시 필요하다
10 @Injectable({providedIn:'root'})
11 export class RankersComponent{
12
13     constructor(private http : HttpClient) {
14     }
15     //타입 스크립트 함수 선언방식 함수이름(매개인자) 리턴형{
16     // get<타입 명시> 로 어떤 json 데이터가 올지 알 수 있다
17     //}
18     getRanking():Observable<Ranker[]>{
19         return this.http.get<Ranker[]>('http://localhost:765/main')
20     }
21 }
22
```

랭커에 관련된 클래스를 생성

Constructor에서  
Http객체를 생성하는 것을 볼 수 있다.

Observable은 시간에 따라 변하는 값  
을 받는 객체<15>

Ranker를 미리 선언해 뒀으므로  
http.get에서 json정보를 알아서  
Ranker class에 대응시켜준다.



# Angular 예시 코드 생성

```
1 import { Component, OnInit } from '@angular/core';
2 import { RankersComponent } from '../rankers/rankers.component';
3 import { Ranker } from '../ranker';
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css']
8 })
9 export class AppComponent implements OnInit {
10   title = 'RankingList';
11   ranking: Ranker[];
12   constructor(private rankingcomponent: RankersComponent) {}
13   ngOnInit() {
14     this.getRankers();
15   }
16   getRankers(): void {
17     this.rankingcomponent.getRanking().subscribe(jsondata => {
18       this.ranking = jsondata;
19       console.log(this.ranking);
20     });
21   }
22 }
```

- 역시나 constructor에서 rankingcomponent를 선언
- ranking 타입 명시
- ngOnInit로 초기화 이때 정보를 불러와 보시다.
- **funcName():리턴type{}**으로 사용
- **Observable은 .subscribe로 읽음**<15>





# Angular 예시 코드 생성

<> app.component.html ✖

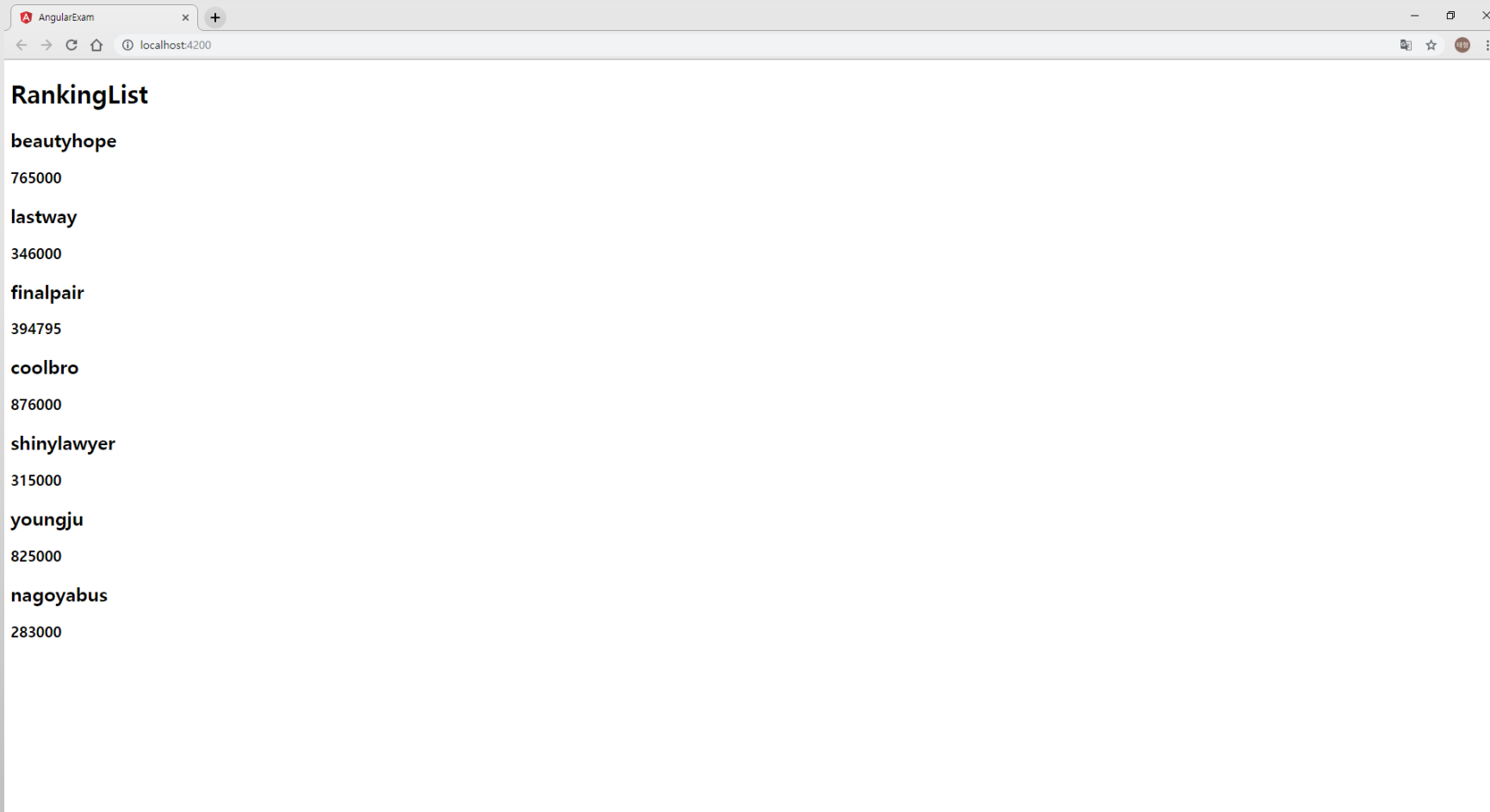
```
1  <!--The content below is only a place
2  <h1>
3  |   {{ title }}
4  </h1>
5  <div *ngFor="let rank of ranking">
6  |   <h2>{{ rank.name }}</h2>
7  |   <h3>{{ rank.score }}</h3>
8  </div>
9
```

- **템플릿과 코드가 분리**되어 있는 모습
- 요소에 접근은 `{{var}}`
- **\*ngfor는 Angular 템플릿에서 반복문**을 사용하게 함
- 원소 원소마다의 이름과 점수를 컴포넌트화 해준다



# Angular 예시 코드 생성 (localhost:4200)

## yarn ng serve --open



# Vue 예시 코드 생성

<17 - 1>

- node init
- yarn add vue-cli
- yarn vue init webpack app-name
- 하나의 파일로 하나의 컴포넌트가 관리가능

```
App.vue x
1 <template>
2   <div id="app">
3     
4     <router-view/>
5   </div>
6 </template>
7
8 <script>
9 export default {
10   name: 'App'
11 }
12 </script>
13
14 <style>
15 #app {
16   font-family: 'Avenir', Helvetica, Arial, sans-serif;
17   -webkit-font-smoothing: antialiased;
18   -moz-osx-font-smoothing: grayscale;
19   text-align: center;
20   color: #2c3e50;
21   margin-top: 60px;
```

## VUEEXAMPLE

- ▾ frontend
  - build
  - config
  - node\_modules
- ▾ src
  - assets
  - components
  - router

### App.vue

JS main.js

▸ static

▸ test

🔗 .babelrc

⚙️ .editorconfig

🔍 .eslintignore

🔍 .eslintrc.js

🔍 .gitignore

JS .postcssrc.js

<> index.html

{ } package.json

📖 README.md

🔗 yarn.lock

▸ node\_modules

{ } package.json

🔗 yarn.lock



# Vue 예시 코드 생성

JS main.js ✕

```
1  // The Vue build version to load
2  // (runtime-only or standalone)
3  import Vue from 'vue'
4  import App from './App'
5  import router from './router'
6  import axios from 'axios'
7  Vue.config.productionTip = false
8  Vue.prototype.$http = axios
9  /* eslint-disable no-new */
10 new Vue({
11   el: '#app',
12   router,
13   components: { App },
14   template: '<App/>'
15 })
16
```

- Vue 객체의 모습
- el로 기존에 있는 요소에 접근 가능
- vue-resource의 문제로 인해 axios를 권장하는 상황<19>
- yarn add axios를 추가



# Vue 예시 코드 생성

```
JS index.js x
1  import Vue from 'vue'
2  import Router from 'vue-router'
3  import HelloWorld from '@/components/HelloWorld'
4
5  Vue.use(Router)
6
7  export default new Router({
8    routes: [
9      {
10       path: '/',
11       name: 'HelloWorld',
12       component: HelloWorld
13     }
14   ]
15 })
16
```

- Vue.use() 함수로 js파일로 작성된 자체 플러그인을 사용할 수 있게 해준다
- HelloWorld Vue파일을 만들고 이를 사용



# Vue 예시 코드 생성

▼ HelloWorld.vue ✕

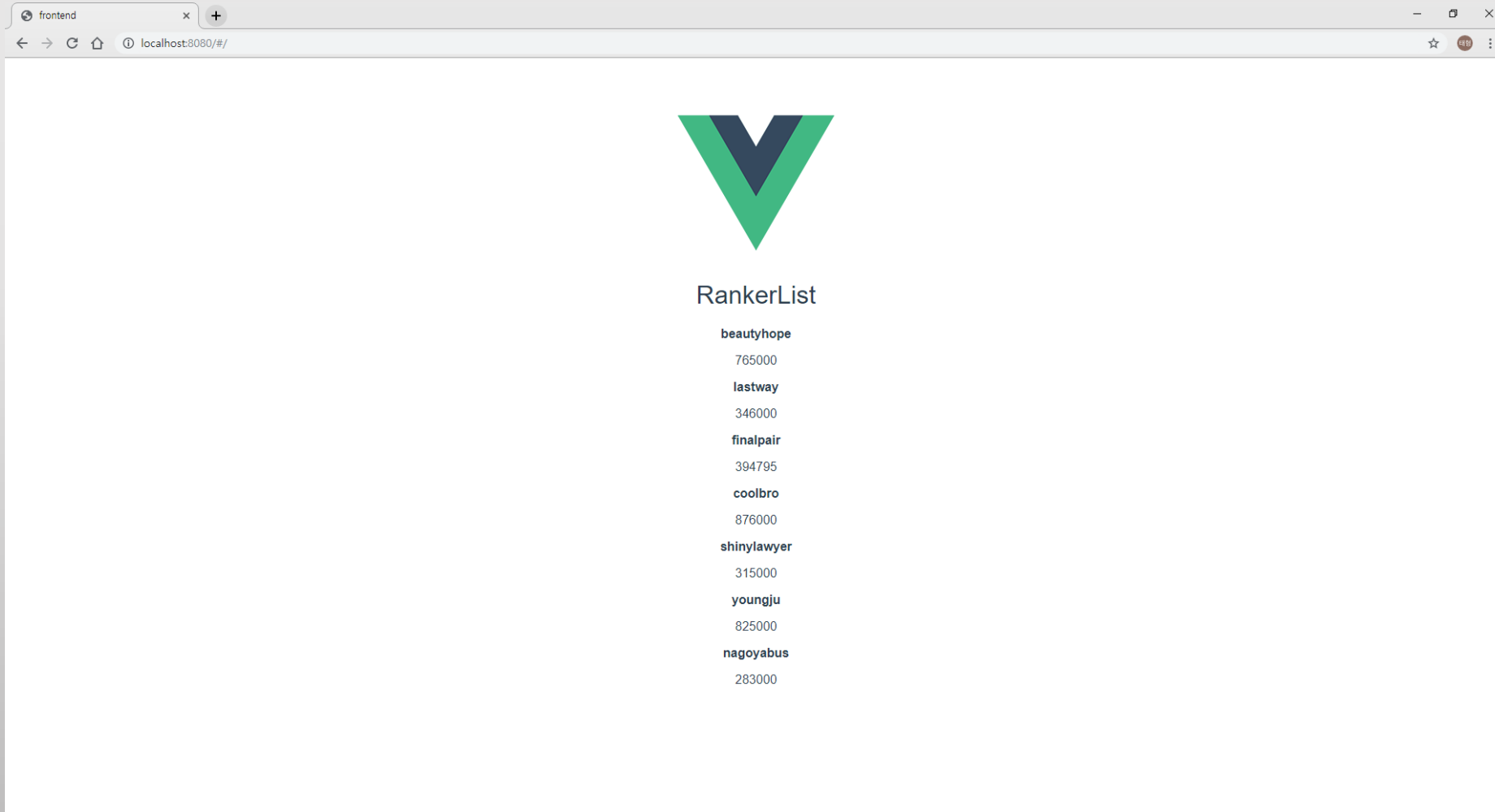
```
1
2 <template>
3   <div class="hello">
4     <h1>RankerList</h1>
5     <div v-for = 'ranker in rankers' :key = "ranker.name"
6       <strong>{{ranker.name}}</strong>
7       <p>{{ranker.score}}</p>
8     </div>
9   </div>
10 </template>
11
12 <script>
13 export default {
14   /* eslint-disable*/
15   name: 'HelloWorld',
16   created () {
17     this.$http.get('http://localhost:765/main')
18       .then((response) => {
19         this.rankers = response.data
20         console.log(this.rankers)
21       })
22   },
23   data () {
24     return {
25       rankers: []
26     }
27   }
28 }
29 </script>
```

- **v- for**로 역시 컴포넌트를 반복해서 생성
- created에서 초기화 작업
- **data()**가 스크립트의 데이터를 가공하여 반환



# Vue 예시 코드 생성 (localhost:8080)

## yarn start





-chapter-

# React/Angular/Vue 비교

상황마다 다른 장점의 비교





# 트렌드 비교

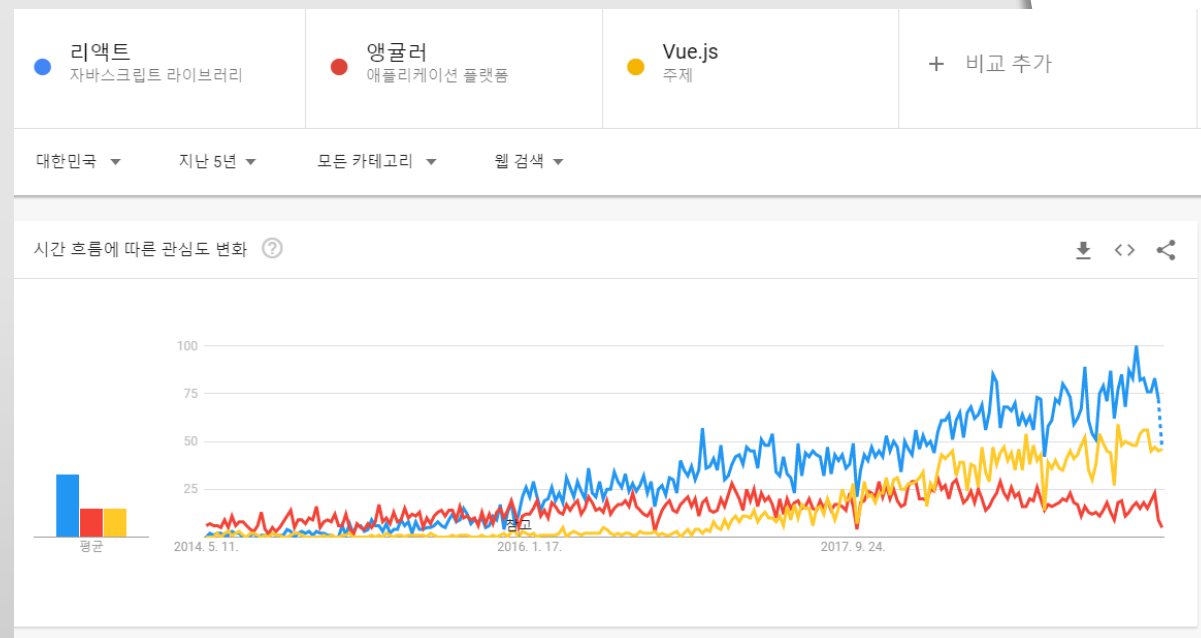
커뮤니티 활성화수준 비교

# 구글 트렌드로 비교

<20>



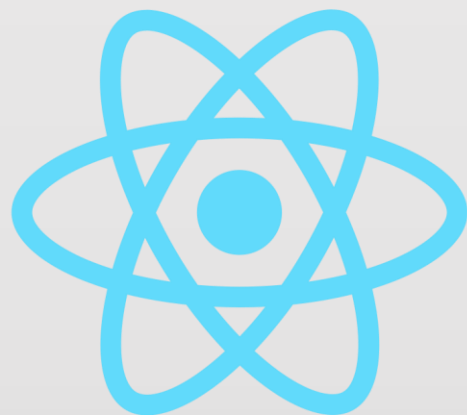
전 세계 기준



대한민국 기준



# 깃 허브에서의 비교



Watch 6,638 Star 128,598 Fork 23,582

<23-1>  
리액트: 128,598star



Watch 3,297 Star 47,855 Fork 12,761

<23-3>  
앵귤러: 47,855star



Watch 5,872 Star 137,901 Fork 19,667

<23-2>  
뷰: 137,901star



# Stack Overflow Trends

<21>

[Stack Overflow Insights](#) > Trends

## Stack Overflow Trends

See how technologies have trended over time based on use of their tags since 2008, when Stack Overflow was founded. Enter up to 15 tags to compare growth and decline.

Tags:

reactjs ×

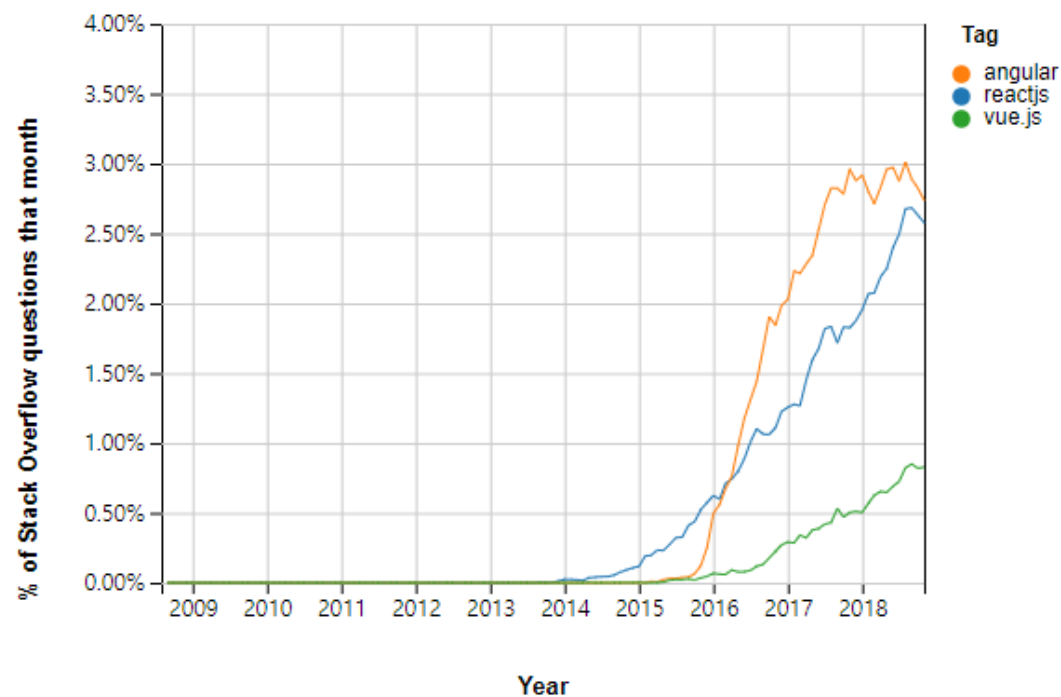
angular ×

vue.js ×

Don't know what tags to look at? Try one of our presets:

- [Most Popular Languages \(TIOBE Index for May 2017\)](#)
- [Operating Systems](#)
- [Mobile Operating Systems](#)
- [Javascript Frameworks](#)
- [Smaller Javascript Frameworks](#)
- [Closed-source Browser Plugins](#)
- [Data Science and Big Data](#)
- [Apache Open-source Projects](#)

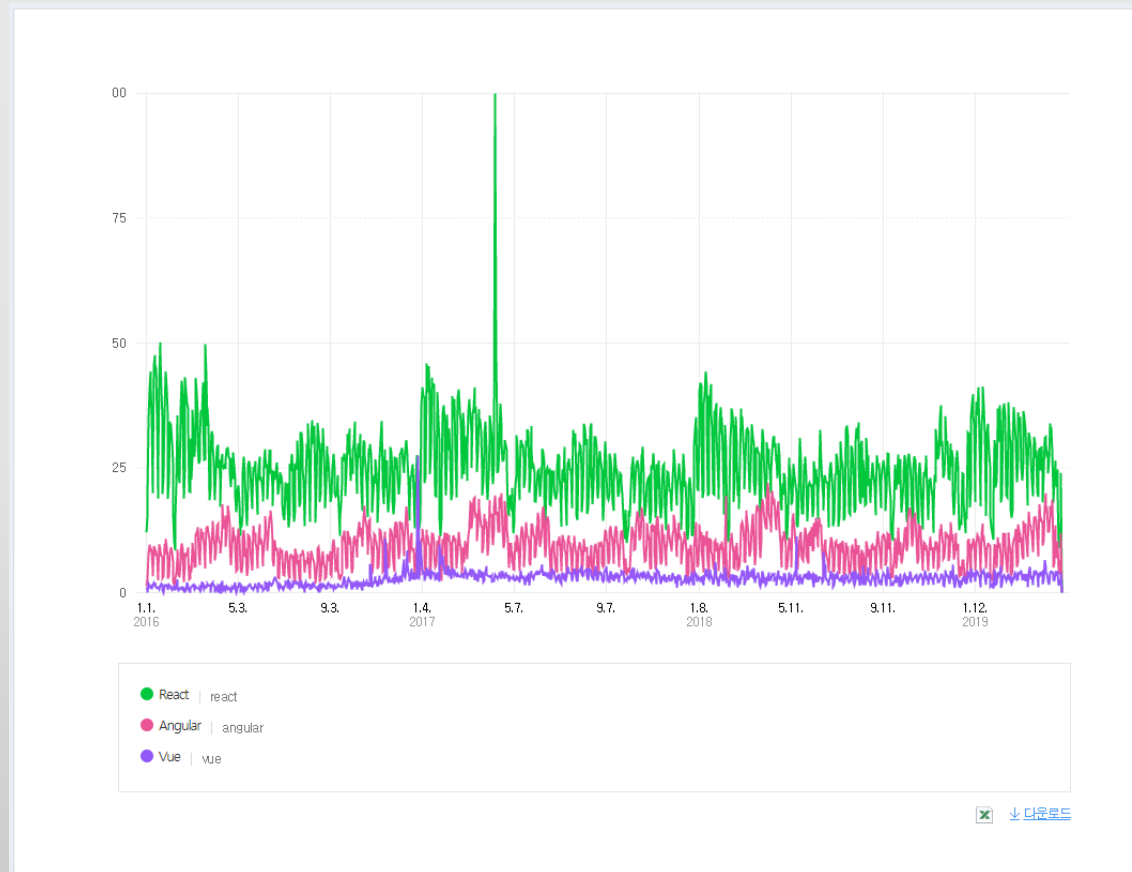
For more on this tool and what you can learn from it, see our [blog post](#).





# 국내 사이트에서의 검색량

<22>



Naver 트렌드(2016년 01월부터)



# 트렌드 비교 결과

- React는 현재에서도 여전히 강세를 보이고 있음
- 외국에서는 Angular도 React못지않게 정말 활발하게 쓰이고 있다
- 국내에서는 React가 압도적으로 많이 쓰이고 있다
- 하지만 최근 Vue가 국내에서 인기가 꽤나 올라간 것을 볼 수 있다.



# 학습 코스트 비교

낮은 학습코스트도 개발과정에서 매우 중요하다



# 왜 어떤거는 쉽고 어떤거는 어려운가

- 결국 모든 프레임워크는 Javascript만 알면 사용하기 어렵다.
- React는 JSX를 추가적으로 배워야 한다.
- Angular는 JS를 아예 사용을 잘 안 한다 대신 Typescript를 사용  
이 자체가 Angular를 매우 어렵게 한다
- Vue가 쉬운 이유 중 하나는 초기 진입단계에서 **추가로 배워야할 요소가 비교적 적기 때문**





# 라이프 사이클

- 동적 렌더링을 지원하고 실시간으로 변하는 데이터를 다루는 이상, 라이프 사이클을 정확히 알아야 데이터를 적시에 업데이트 하고 적용시킬 수 있음
- 라이프 사이클에 대한 깊은 이해는 코드 최적화의 수준을 상당히 높임

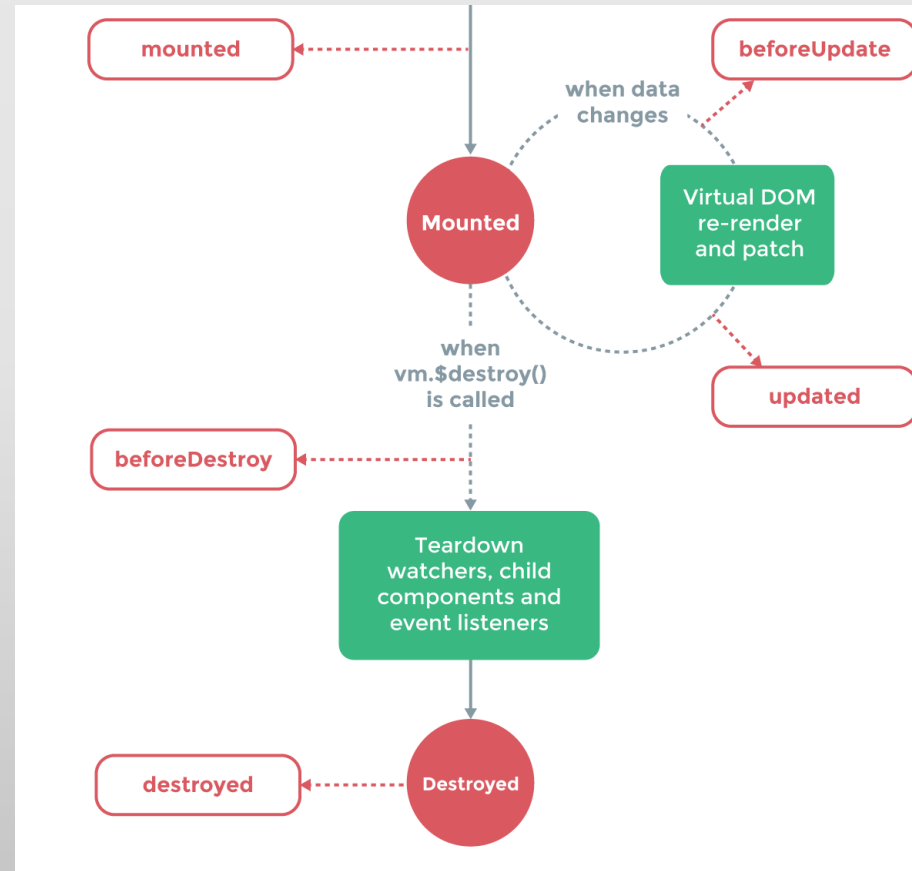
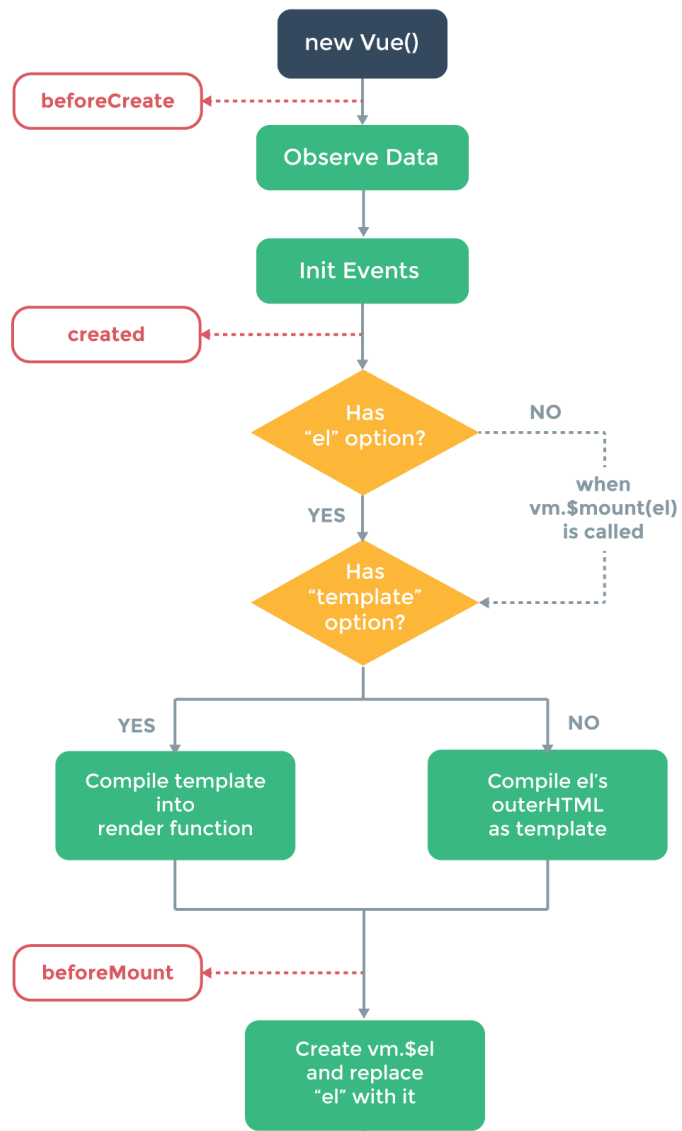


# 라이프 사이클

Angular	React
constructor	constructor()
ngOnChanges	componentWillMount()
ngOnInit	render()
ngDoCheck	componentDidMount()
ngAfterContentInit	componentWillReceiveProps()
ngAfterContentChecked	shouldComponentUpdate()
ngAfterViewInit	componentWillUpdate()
ngAfterViewChecked	render()
ngOnDestroy	componentDidUpdate()
	componentWillUnmount()

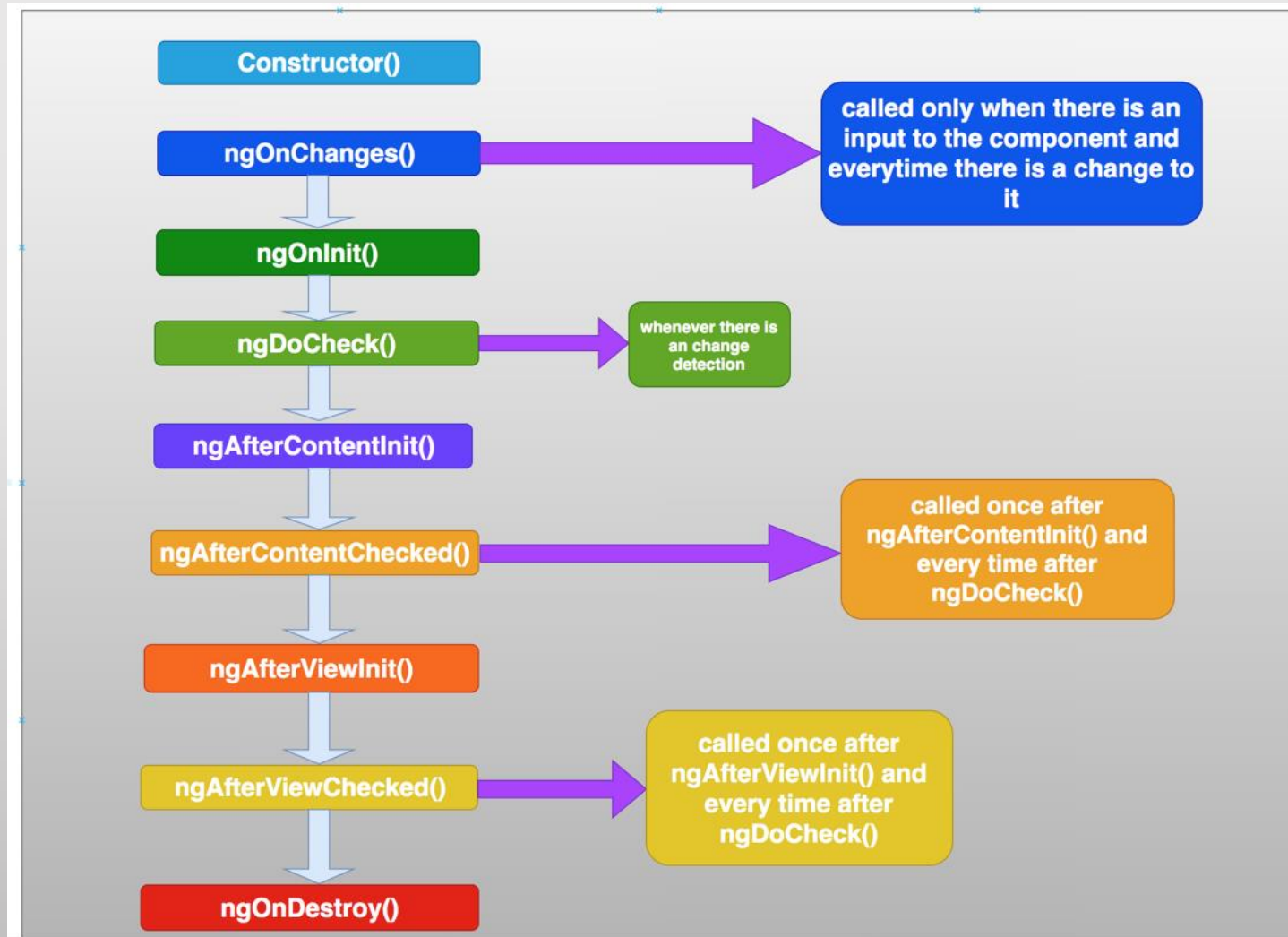
- Vue
- 1. beforeCreate
  - 2. created
  - 3. beforeMount
  - 4. mounted
  - 5. beforeUpdate
  - 6. updated
  - 7. beforeDestroy
  - 8. destroyed
- <24>

# Vue 라이프 사이클



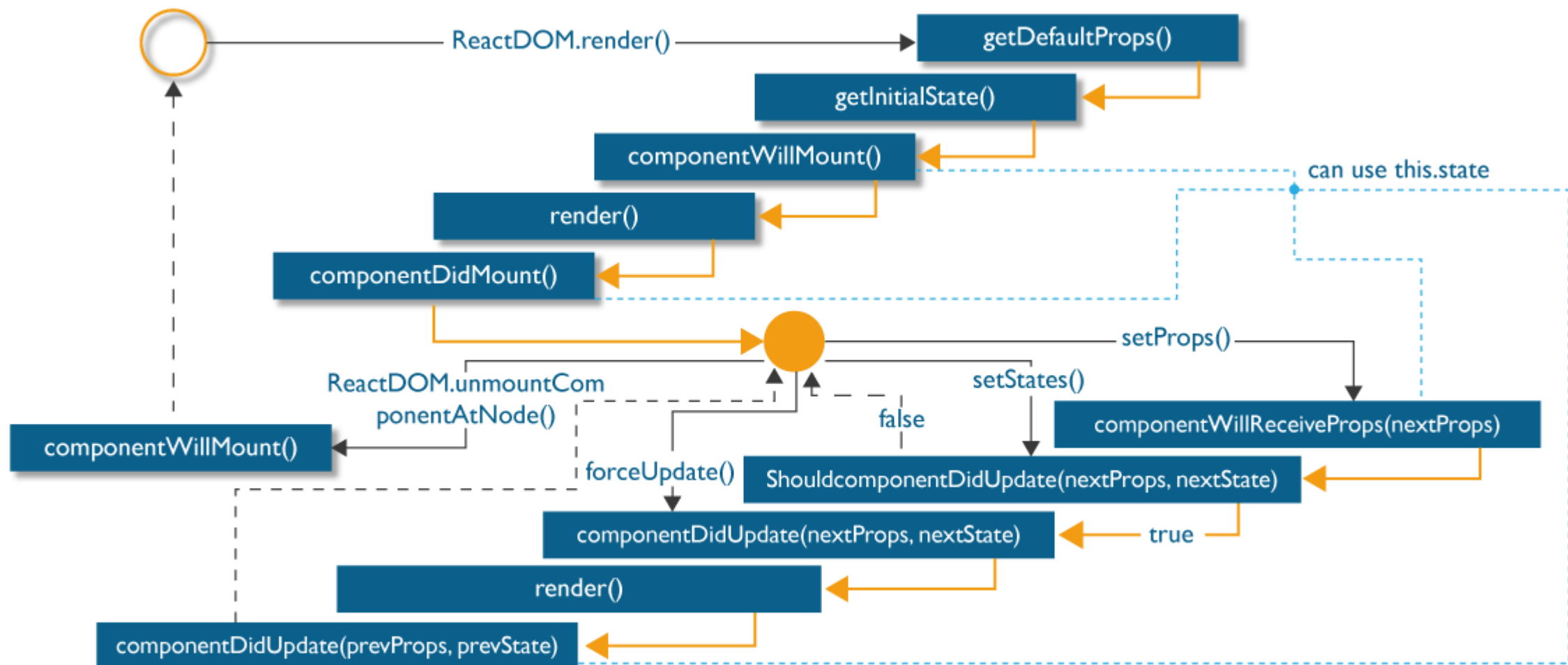


# Angular 라이프 사이클





# React 라이프 사이클





# 라이프 사이클

- Vue나 Angular는 꽤 간단한 구조의 라이프 사이클을 지원
- 하지만 React는 라이프 사이클이 상당히 복잡
- 만약 Angular의 라이프 사이클에 익숙하다면 Vue의 사이클의 적응도 상당히 쉬울 것



# 성능 비교

속도와 기능제공기준



# 기본적인 성능 비교

Duration in milliseconds  $\pm$  standard deviation    Memory allocation in MBs  $\pm$  standard deviation

Name	angular-v6.1.0-keyed	angular-v6.1.0-non-keyed	react-v16.4.1-keyed	react-v16.4.1-non-keyed	vue-v2.5.16-keyed	vue-v2.5.16-non-keyed
<b>create rows</b> Duration for creating 1000 rows after the page loaded.	185.2 $\pm$ 10.2 (1.0)	177.9 $\pm$ 7.7 (1.0)	180.5 $\pm$ 7.3 (1.0)	184.7 $\pm$ 7.4 (1.0)	182.1 $\pm$ 7.6 (1.0)	182.0 $\pm$ 8.6 (1.0)
<b>replace all rows</b> Duration for updating all 1000 rows of the table (with 5 warmup iterations).	161.2 $\pm$ 2.7 (2.7)	59.2 $\pm$ 2.6 (1.0)	157.3 $\pm$ 2.0 (2.7)	61.4 $\pm$ 2.6 (1.0)	158.8 $\pm$ 2.7 (2.7)	67.2 $\pm$ 2.1 (1.1)
<b>partial update</b> Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows.	68.8 $\pm$ 3.7 (1.0)	66.4 $\pm$ 2.6 (1.0)	61.9 $\pm$ 2.7 (1.2)	61.3 $\pm$ 2.4 (1.2)	156.4 $\pm$ 9.8 (2.4)	169.7 $\pm$ 23.4 (2.6)
<b>select row</b> Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	7.9 $\pm$ 4.3 (1.0)	7.0 $\pm$ 3.0 (1.0)	10.3 $\pm$ 2.1 (1.0)	10.4 $\pm$ 2.3 (1.0)	10.6 $\pm$ 2.0 (1.0)	11.2 $\pm$ 2.8 (1.0)
<b>swap rows</b> Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	105.8 $\pm$ 1.8 (6.6)	13.5 $\pm$ 4.4 (1.0)	106.5 $\pm$ 1.9 (6.7)	14.8 $\pm$ 4.5 (1.0)	20.0 $\pm$ 2.9 (1.3)	13.8 $\pm$ 1.5 (1.0)
<b>remove row</b> Duration to remove a row. (with 5 warmup iterations).	47.1 $\pm$ 3.0 (1.5)	31.8 $\pm$ 3.3 (1.0)	49.6 $\pm$ 0.8 (1.6)	37.8 $\pm$ 1.5 (1.2)	54.2 $\pm$ 2.2 (1.7)	38.3 $\pm$ 1.5 (1.2)
<b>create many rows</b> Duration to create 10,000 rows	1,693.9 $\pm$ 70.1 (1.1)	1,647.2 $\pm$ 47.6 (1.0)	1,935.4 $\pm$ 33.6 (1.2)	1,945.2 $\pm$ 24.4 (1.2)	1,603.2 $\pm$ 34.8 (1.0)	1,581.5 $\pm$ 48.4 (1.0)
<b>append rows to large table</b> Duration for adding 1000 rows on a table of 10,000 rows.	243.3 $\pm$ 6.3 (1.0)	243.8 $\pm$ 5.1 (1.0)	268.6 $\pm$ 6.9 (1.1)	267.8 $\pm$ 5.5 (1.1)	342.5 $\pm$ 6.0 (1.4)	350.2 $\pm$ 13.8 (1.4)
<b>clear rows</b> Duration to clear the table filled with 10,000 rows.	263.9 $\pm$ 3.0 (1.5)	257.4 $\pm$ 2.2 (1.5)	175.4 $\pm$ 4.1 (1.0)	175.3 $\pm$ 1.6 (1.0)	191.9 $\pm$ 6.1 (1.1)	192.0 $\pm$ 9.5 (1.1)
<b>slowdown geometric mean</b>	1.53	1.05	1.53	1.09	1.41	1.21

Name	angular-v6.1.0-keyed	angular-v6.1.0-non-keyed	react-v16.4.1-keyed	react-v16.4.1-non-keyed	vue-v2.5.16-keyed	vue-v2.5.16-non-keyed
<b>ready memory</b> Memory usage after page load.	6.0 $\pm$ 0.0 (2.2)	6.0 $\pm$ 0.1 (2.2)	2.8 $\pm$ 0.2 (1.0)	2.8 $\pm$ 0.2 (1.0)	2.7 $\pm$ 0.2 (1.0)	2.7 $\pm$ 0.2 (1.0)
<b>run memory</b> Memory usage after adding 1000 rows.	9.8 $\pm$ 0.0 (1.5)	9.8 $\pm$ 0.0 (1.5)	6.7 $\pm$ 0.0 (1.0)	6.7 $\pm$ 0.0 (1.0)	7.1 $\pm$ 0.0 (1.1)	7.1 $\pm$ 0.0 (1.1)
<b>update each 10th row for 1k rows (5 cycles)</b> Memory usage after clicking update every 10th row 5 times	9.8 $\pm$ 0.0 (1.4)	9.8 $\pm$ 0.0 (1.4)	7.6 $\pm$ 0.0 (1.1)	7.6 $\pm$ 0.0 (1.1)	7.2 $\pm$ 0.0 (1.0)	7.1 $\pm$ 0.0 (1.0)
<b>replace 1k rows (5 cycles)</b> Memory usage after clicking create 1000 rows 5 times	10.1 $\pm$ 0.0 (1.4)	9.8 $\pm$ 0.0 (1.4)	7.9 $\pm$ 0.0 (1.1)	10.8 $\pm$ 0.1 (1.5)	7.2 $\pm$ 0.0 (1.0)	7.1 $\pm$ 0.0 (1.0)
<b>creating/clearing 1k rows (5 cycles)</b> Memory usage after creating and clearing 1000 rows 5 times	6.4 $\pm$ 0.0 (2.1)	6.4 $\pm$ 0.0 (2.1)	3.8 $\pm$ 0.0 (1.3)	3.8 $\pm$ 0.0 (1.3)	3.0 $\pm$ 0.0 (1.0)	3.0 $\pm$ 0.0 (1.0)

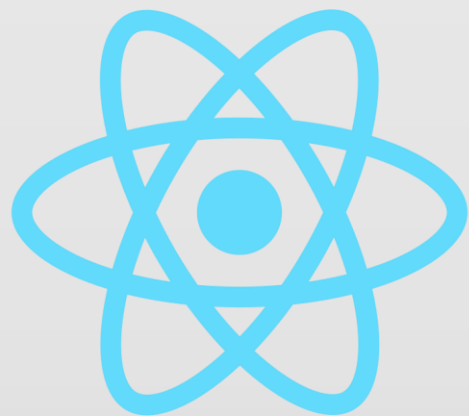
<26>

**Vue가** 데이터 생성과 메모리 접근면에서 React나 Angular보다 **좋은 모습을** (2018년 9월기준)





# 조기 컴파일(Ahead of Time)



React와 Vue는 JS코드가 VirtualDOM을 한번 거쳐야 하기 때문에 AOT를 지원하지 않는다.

<27>

AOT를 지원



# 조기 컴파일(Ahead of Time)

- Angular는 Virtual DOM에서 오는 이득을 볼 수 없음
- Angular는 AOT컴파일을 옵션적으로 지원  
디폴트로 지원하지 않는다
- Angular 기준 속도가 약 50% 향상된다



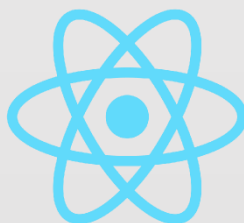
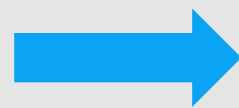
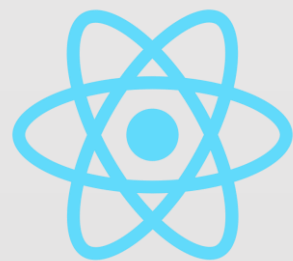
# Native 앱 개발

<28>

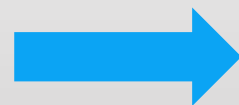
- 브라우저 상에서 보는 것이 아니라 컴퓨터나 스마트폰에 앱을 설치 후 볼 수 있게 해주는 도구
- Android/IOS버전을 동시에 개발할 수 있게 해준다
- 웹 콘텐츠로의 확장을 용이하게 해줌
- 그러나 유지 보수가 어렵다
- 아직 시험단계라 커뮤니티가 많이 활성화 되어 있지 않다.



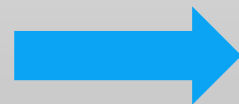
# 모바일 Native



React Native <43>



<44>



<45>



# 데스크탑 Native



<46>

현재 Electron 에서 세가지의 모든 프레임워크방식을 지원

# 3

-chapter-

## 결론 및 추가정보

요약 그리고 더 알아 볼만한 요소들



# 요약

- React  
커뮤니티가 제일 활성화 되어 있으며, JSX의 사용으로 코드의 가독성이 뛰어남
- Angular  
two way binding의 장점을 사용할 수 있으며, 템플릿과 코드의 분리가 가능해 추후의 유지보수가 원활
- Vue  
쉽게 배울 수 있어 빠르게 개발에 참여가 가능하며, 1개의 뷰파일로 컴포넌트의 모든 특성을 담을 수 있는 점이 특징



# 개인 소견

- React  
밸런스가 제일 좋은 무난한 프레임워크라고 생각합니다.  
또 추후에 모바일 앱이나 데스크톱 앱을 개발할 경우가 생긴다면 추천
- Angular  
대규모 프로젝트와 협업이 중요시되는 상황에서 추천
- Vue  
비교적 작은 프로젝트를 빠르게 가동시켜야 할 때 추천





# 주목받고 있는 새로운 프레임워크들

- Next.js <29>  
create-react-app과 비슷하게 환경설정을 자동으로 구성해준다  
react와 javascript를 기반으로 하여 친숙  
Hot Module Replacement로 파일이 변경돼도 리로딩 없이 적용
- Nuxt.js <30>  
마찬가지로 서버사이드 렌더링을 도와주면서 Vue기반이기 때문에 사용하기 쉽다  
역시 Reload 없이 페이지의 변화를 보여줌

# // 출처 // (내용)

<0>: Deep dive into Modern frameworks - HTML5 Forum 2018

-프레임워크의 필요성-

<1>: 나무위키(프론트엔드/프레임워크)

<https://namu.wiki/w/%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC>

<https://namu.wiki/w/%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%94%EB%93%9C>

<2>: [개발용어] 라이브러리, 프레임워크, 아키텍처, 플랫폼이란?

<https://blog.gaerae.com/2016/11/what-is-library-and-framework-and-architecture-and-platform.html>

<3>: Angular2 특징 및 소개

<https://dalkomit.tistory.com/156>

<4>: 제이쿼리는 필요 없을 수도 있다?

<https://www.zerocho.com/category/jQuery/post/57b356d4d841141500b31e1e>

<5>: [JavaScript] DOM이란 무엇인가?

<https://m.blog.naver.com/magnking/220972680805>

<6>: React: #1 React의 탄생배경과 특징

<https://medium.com/@RianCommunity/react%EC%9D%98-%ED%83%84%EC%83%9D%EB%B0%B0%EA%B2%BD%EA%B3%BC-%ED%8A%B9%EC%A7%95-4190d47a28f>

<7>: React JS(리액트) Angular JS(앵귤러) 차이 및 비교

<https://jayzzz.tistory.com/60>

# // 출처 // (내용)

<8>: [번역] 리액트에 대해서 그 누구도 제대로 설명하기 어려운 것 - 왜 Virtual DOM 인가?

<https://velopert.com/3236>

<9>: Vue.js 사용하는 이유 :: 마이구미

<https://mygumi.tistory.com/206>

<10>: React보다 Angular2에 더 주목해야하는 이유

<http://sculove.github.io/blog/2016/07/11/reasons-angular2-than-react/>

<11>: [자바스크립트] 왜 JQUERY를 사용하는가?

<https://unikys.tistory.com/300>

<12>: 자바스크립트는 어떻게 작동하는가: V8 엔진의 내부 + 최적화된 코드를 작성을 위한 다섯 가지 팁

<https://engineering.huisseoul.com/%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8%EB%8A%94-%EC%96%B4%EB%96%BB%EA%B2%8C-%EC%9E%91%EB%8F%99%ED%95%98%EB%8A%94%EA%B0%80-v8-%EC%97%94%EC%A7%84%EC%9D%98-%EB%82%B4%EB%B6%80-%EC%B5%9C%EC%A0%81%ED%99%94%EB%90%9C-%EC%BD%94%EB%93%9C%EB%A5%BC-%EC%9E%91%EC%84%B1%EC%9D%84-%EC%9C%84%ED%95%9C-%EB%8B%A4%EC%84%AF-%EA%B0%80%EC%A7%80-%ED%8C%81-6c6f9832c1d9>

# // 출처 // (내용)

-React/Angular/Vue의 개요-

<13>: React/Angular/Vue 홈페이지

<https://reactjs.org/versions/>

<https://news.vuejs.org/>

<https://angular.kr/guide/releases>

<14>: 다른 프레임워크와의 비교

<https://kr.vuejs.org/v2/guide/comparison.html>

-React/Angular/Vue의 구현 방식-

<15>: (번역) Angular의 observable을 이해하고, 생성하고, 구독해보기

[https://feel5ny.github.io/2018/03/25/angular\\_observable/](https://feel5ny.github.io/2018/03/25/angular_observable/)

<16>: [Node.js] express와 vue.js 환경설정

<https://m.blog.naver.com/kangminser88/221146020394>

<17>: Angular/Vue 홈페이지

<https://kr.vuejs.org/v2/guide/installation.html#CLI>

<https://angular.kr/guide/quickstart#install-cli>

<18>: 누구든지 하는 리액트 4편: props 와 state

<https://velopert.com/3629>

<19>: HTTP 요청을 위한 axios

<http://vuejs.kr/update/2017/01/04/http-request-with-axios/>

# // 출처 // (내용)

## -트렌드 비교-

<20>: Google Trends

<https://trends.google.com>

<21>: Stack Overflow Trends

<https://insights.stackoverflow.com/trends>

<22>: 네이버 검색어트렌드

<https://datalab.naver.com/keyword/trendSearch.naver>

<23>: react/vue/angular github

<https://github.com/facebook/react>

<https://github.com/vuejs/vue>

<https://github.com/angular/angular>

## -학습 코스트 비교-

<24>: Vue 홈페이지

<https://kr.vuejs.org/v2/guide/instance.html#%EB%9D%BC%EC%9D%B4%ED%94%84%EC%82%AC%EC%9D%B4%ED%81%B4-%EB%8B%A4%EC%9D%B4%EC%96%B4%EA%B7%B8%EB%9E%A8>

<25>: [앵글러] 라이프사이클 이해하기

<https://shlee1353.github.io/2017/11/15/Angular-Lifecycle-Hooks/>

# // 출처 // (내용)

## -성능 비교-

<26>: Results for js web frameworks benchmark - round 8

<https://stefankrause.net/js-frameworks-benchmark8/table.htm>

<27>: Angular vs. React: Compilers

<https://medium.jonasbandi.net/angular-vs-react-compilers-45b279a8f571>

<28>: [RN] React-Native의 장단점은?

<https://medium.com/@jang.wangsu/rn-react-native%EC%9D%98-%EC%9E%A5%EB%8B%A8%EC%A0%90%EC%9D%80-6e8a2396eea1>

## -결론 및 추가정보-

<29>: [nuxt] nuxt.js 처음부터 시작하기

<http://blog.naver.com/PostView.nhn?blogId=pjt3591oo&logNo=221027685707>

<30>: Next.js 튜토리얼 1편: 시작하기

<https://brunch.co.kr/@hee072794/81>

# // 출처 // (그림)

## -프레임워크의 필요성-

<31>: <https://medium.com/indianic/spring-framework-helps-you-to-develop-robust-java-applications-very-easily-and-very-rapidly-4a29c284996b>

<32>: [https://medium.com/@psychet\\_learn/django-python-django%EB%9E%80-d1165b8e640b](https://medium.com/@psychet_learn/django-python-django%EB%9E%80-d1165b8e640b)

<33>: <https://www.howtogeek.com/253588/what-is-the-microsoft-net-framework-and-why-is-it-installed-on-my-pc/>

<34>: [http://www.hanbit.co.kr/store/books/look.php?p\\_code=B1512579995](http://www.hanbit.co.kr/store/books/look.php?p_code=B1512579995)

<35>: <http://www.nextree.co.kr/p9747/>

<36>: <https://github.com/nkiateam/angular-tutorial/wiki/Angular2-%EC%A0%95%EB%A6%AC>

## -React/Angular/Vue의 개요-

<37>: <https://auth0.com/blog/bootstrapping-a-react-project/>

<38>: <https://egghead.io/courses/get-started-with-angular>

<39>: <https://kr.vuejs.org/v2/guide/index.html>

# // 출처 // (그림)

-React/Angular/Vue비교-

<40>: <https://kr.vuejs.org/v2/guide/instance.html>

<41>: <https://medium.com/bhargav-bachina-angular-training/angular-understanding-angular-lifecycle-hooks-with-a-sample-project-375a61882478>

<42>: <https://www.edureka.co/blog/react-components/>

<43>: <https://www.braze.com/product/alloys/partners/react-native>

<44>: <https://blog.championswimmer.in/2018/04/getting-started-with-weex/>

<45>: <https://pointdeveloper.com/what-is-ngcordova-how-to-use-it-with-ionic//>

<46>: [https://en.wikipedia.org/wiki/Electron\\_\(software\\_framework\)](https://en.wikipedia.org/wiki/Electron_(software_framework))





감사합니다.

