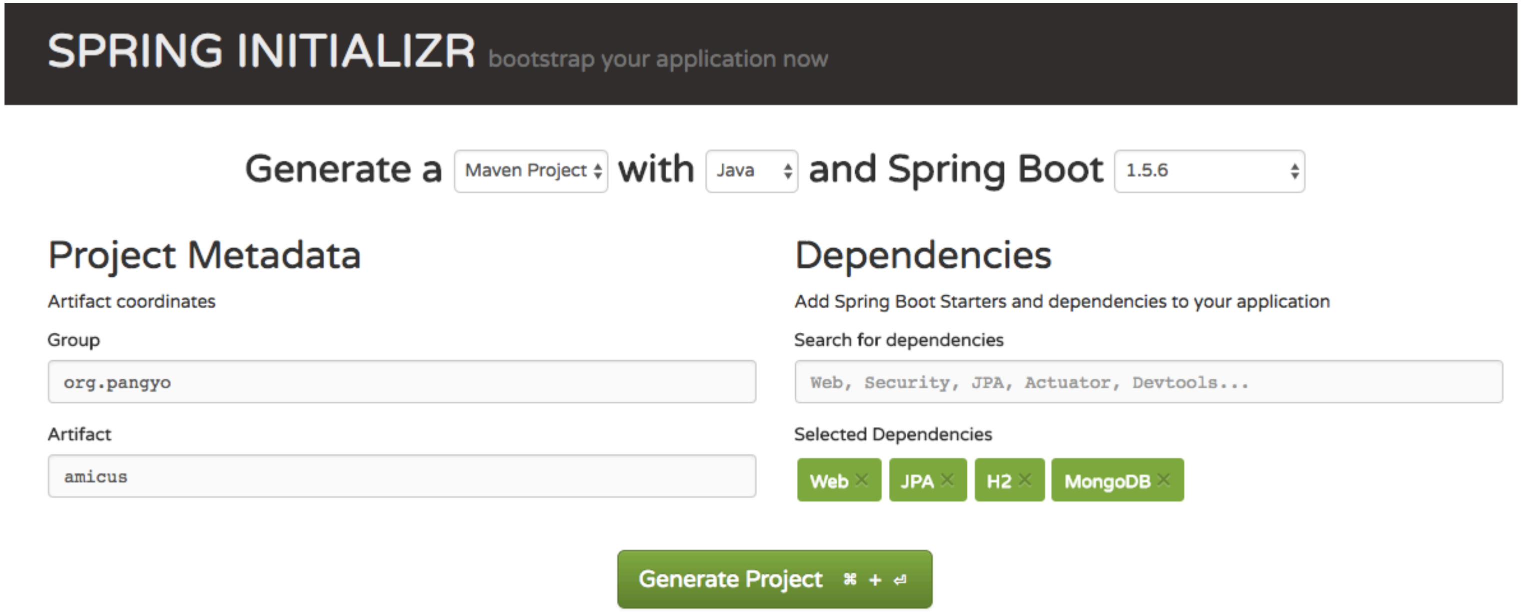
Spring과 React

Init

- http://start.spring.io/ 에 접속하여 spring boot init project를 만듦.
- H2랑 Mongo는 걍 쓸수도 있을거 같아서 넣어 놈.



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 1.5.6

Project Metadata

Artifact coordinates

Group: org.pangyo

Artifact: amicus

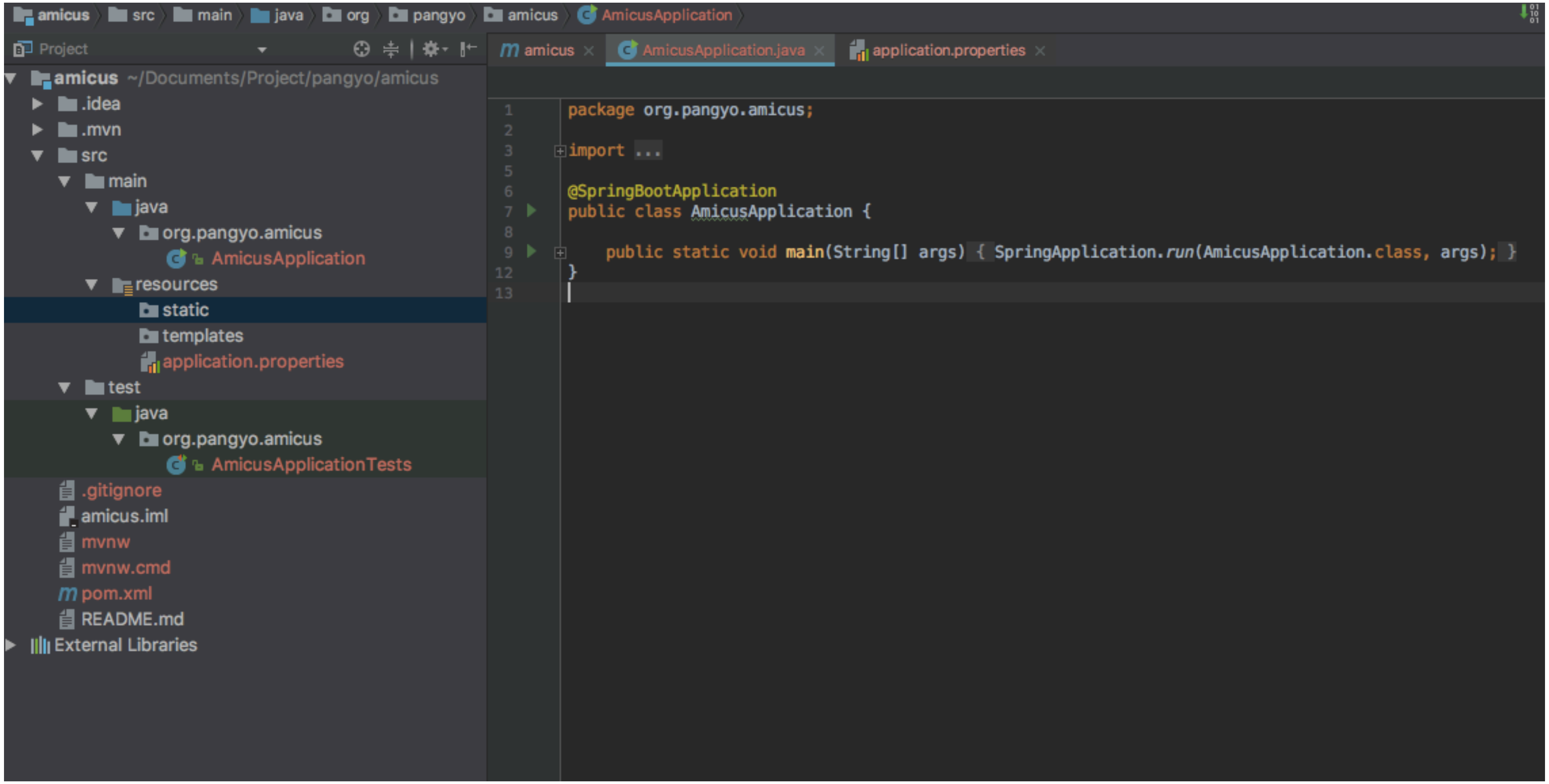
Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

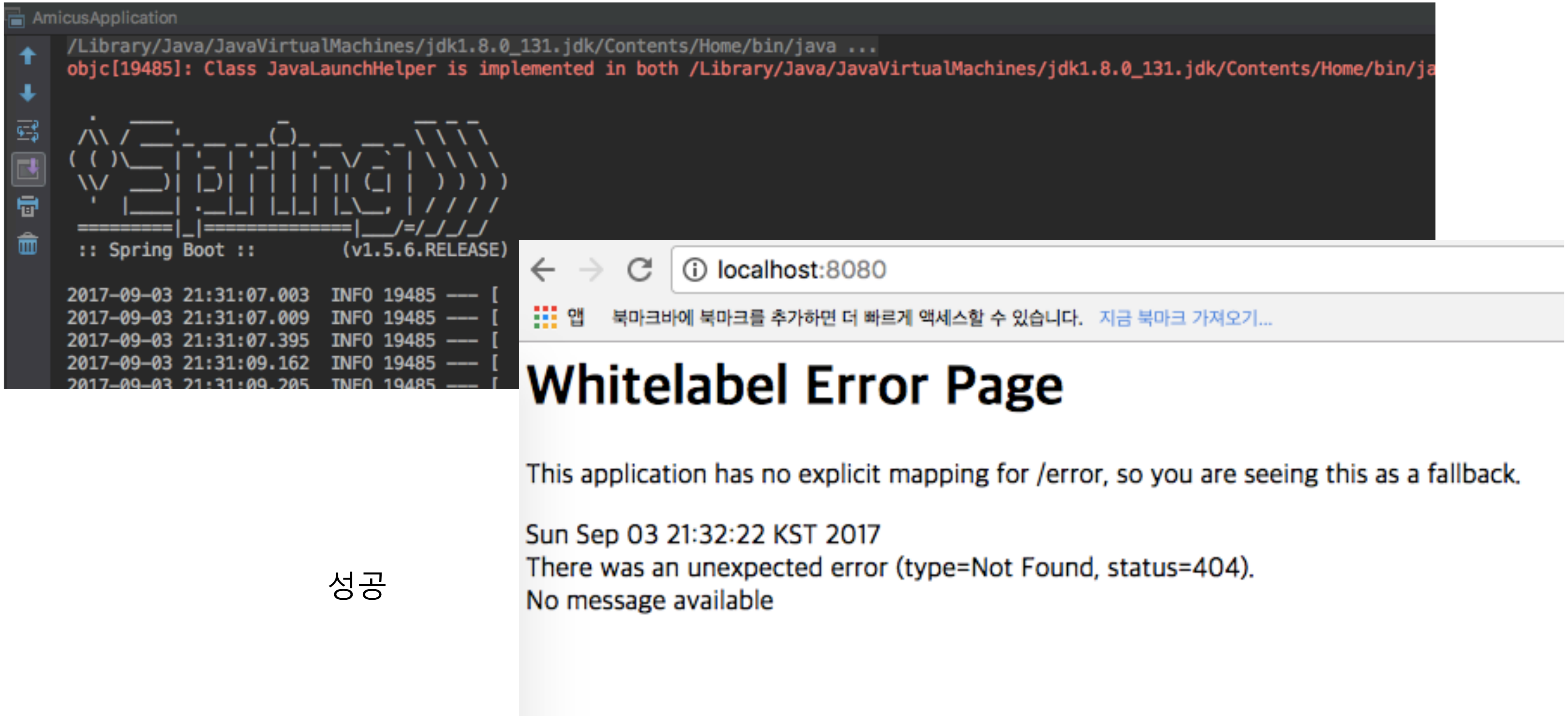
Selected Dependencies: Web, JPA, H2, MongoDB

프로젝트 셋업



```java
package org.pangyo.amicus;

import ...

@SpringBootApplication
public class AmicusApplication {

    public static void main(String[] args) { SpringApplication.run(AmicusApplication.class, args); }
}
```

실행



```
/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java ...
objc[19485]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/ja

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.6.RELEASE)

2017-09-03 21:31:07.003  INFO 19485 --- [
2017-09-03 21:31:07.009  INFO 19485 --- [
2017-09-03 21:31:07.395  INFO 19485 --- [
2017-09-03 21:31:09.162  INFO 19485 --- [
2017-09-03 21:31:09.205  INFO 19485 --- [
```

localhost:8080
앱
북마크바에 북마크를 추가하면 더 빠르게 액세스할 수 있습니다.
지금 북마크 가져오기...

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun Sep 03 21:32:22 KST 2017
There was an unexpected error (type=Not Found, status=404).
No message available

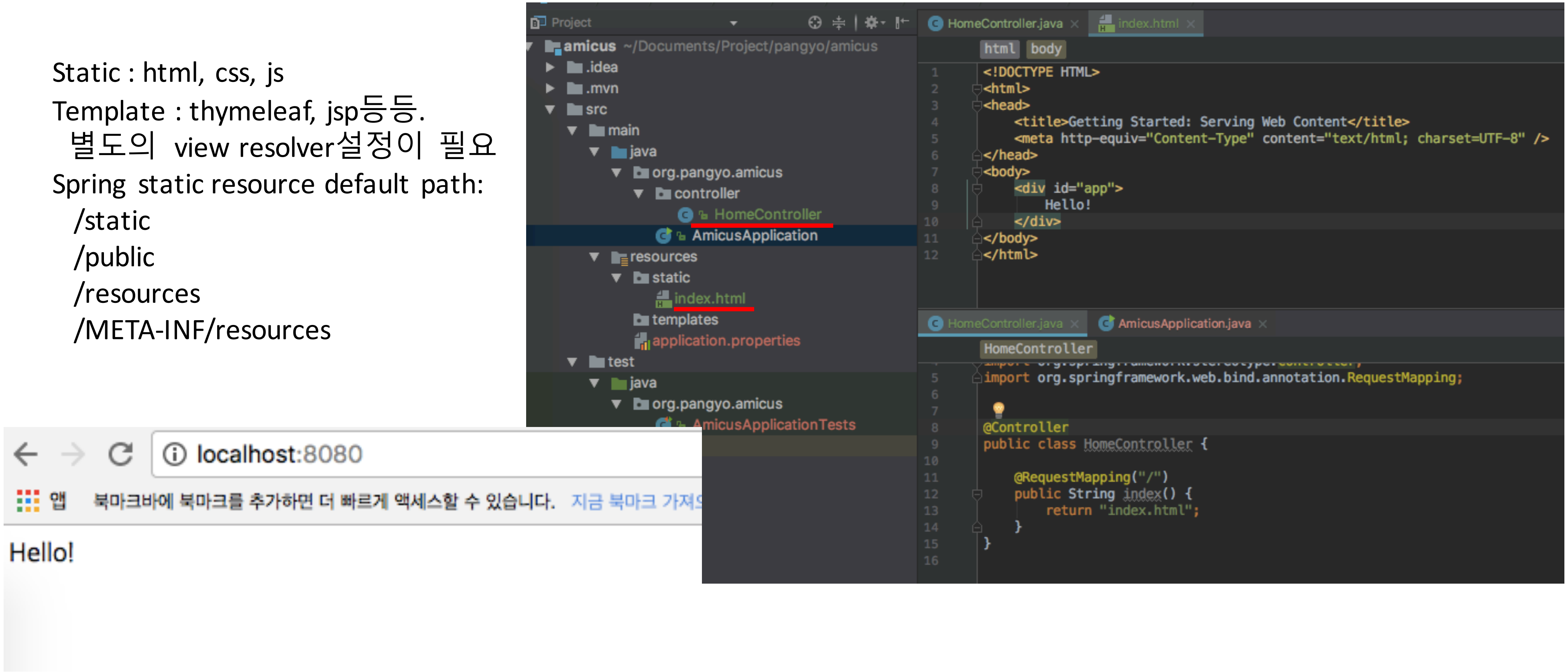
성공

컨트롤러와 기본 페이지 등록

Static : html, css, js

Template : thymeleaf, jsp등등.
  별도의 view resolver설정이 필요

Spring static resource default path:
  /static
  /public
  /resources
  /META-INF/resources

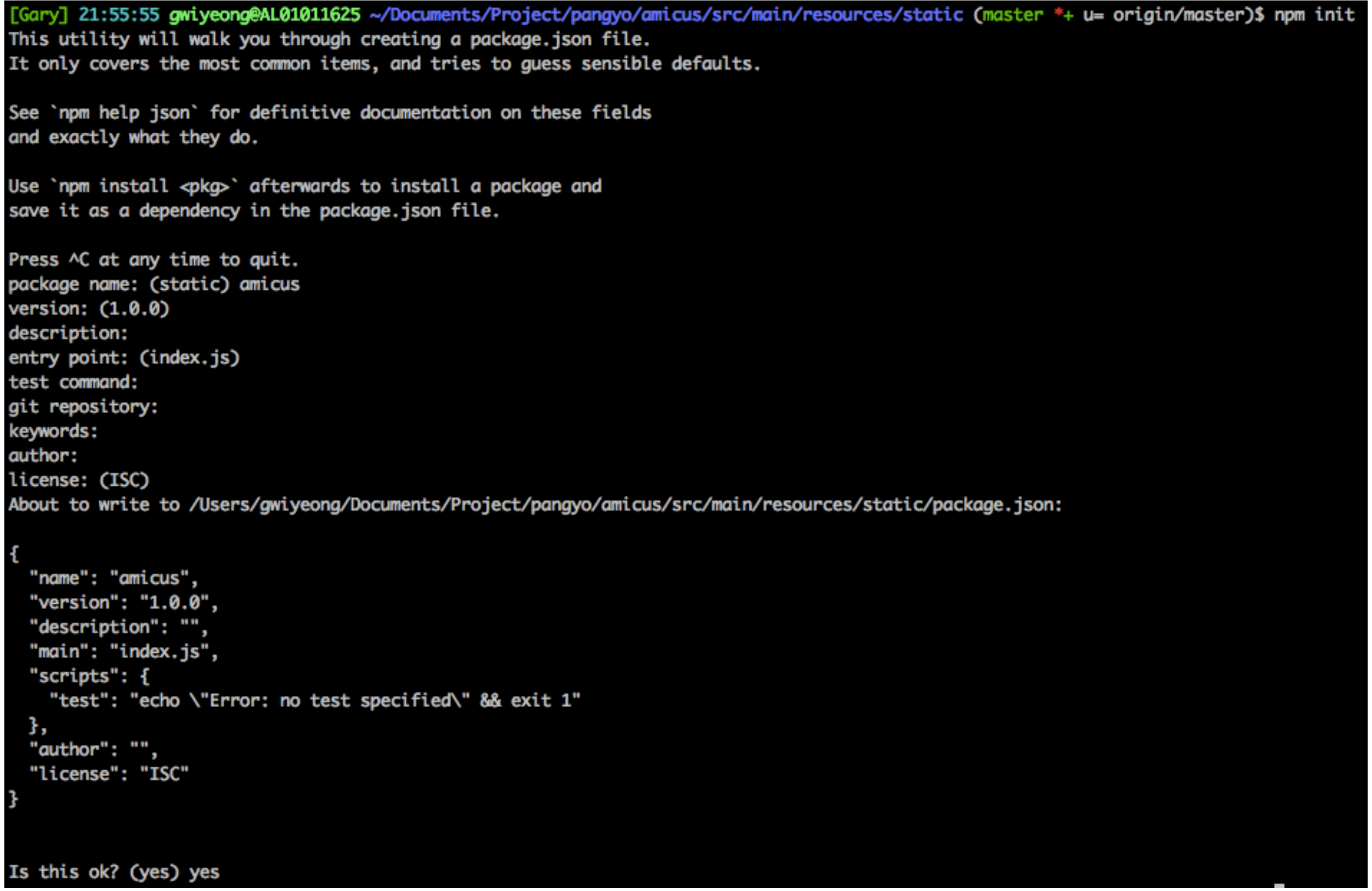


```
<!DOCTYPE HTML>
<html>
<head>
    <title>Getting Started: Serving Web Content</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <div id="app">
        Hello!
    </div>
</body>
</html>
```

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;


@Controller
public class HomeController {

    @RequestMapping("/")
    public String index() {
        return "index.html";
    }
}
```

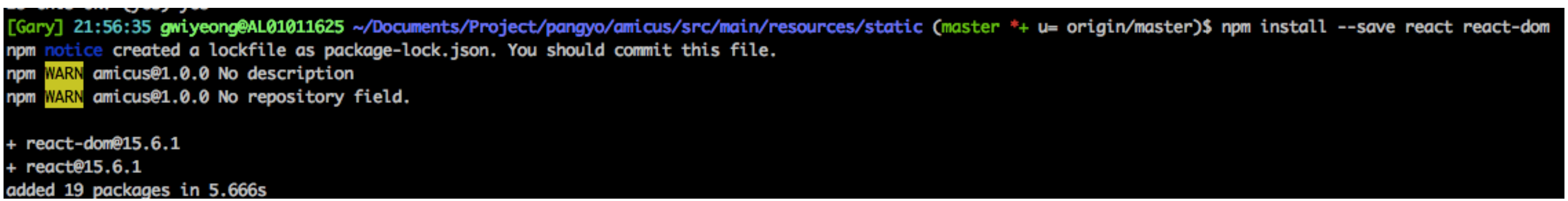
Hello!

React 설치 – npm init



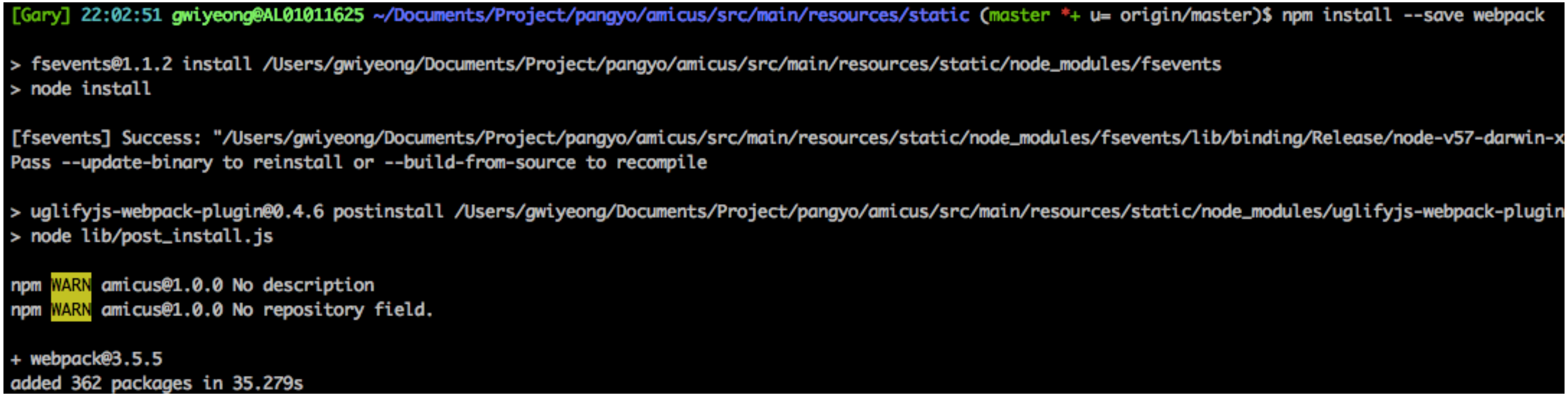
```
[Gary] 21:55:55 gwiyeong@AL01011625 ~/Documents/Project/pangyo/amicus/src/main/resources/static (master *+ u= origin/master)$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (static) amicus
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /Users/gwiyeong/Documents/Project/pangyo/amicus/src/main/resources/static/package.json:

{
  "name": "amicus",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this ok? (yes) yes
```

React 설치 – react 설치



```
[Gary] 21:56:35 gwiyeong@AL01011625 ~/Documents/Project/pangyo/amicus/src/main/resources/static (master *+ u= origin/master)$ npm install --save react react-dom
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN amicus@1.0.0 No description
npm WARN amicus@1.0.0 No repository field.

+ react-dom@15.6.1
+ react@15.6.1
added 19 packages in 5.666s
```

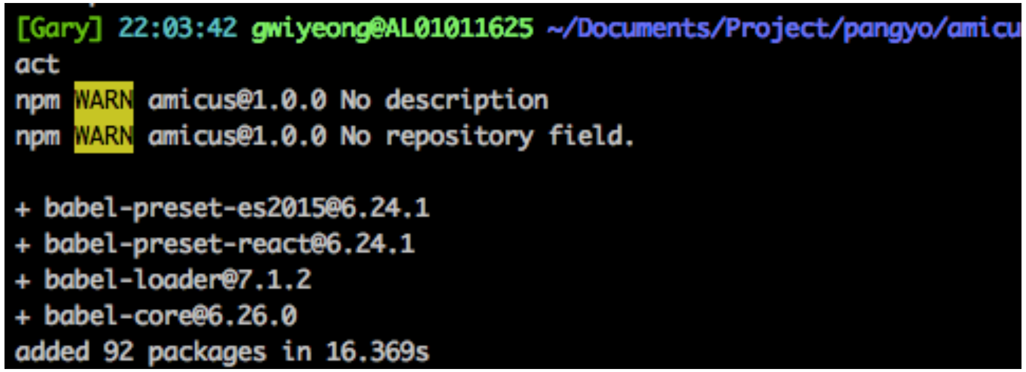
React 설치 – webpack 설치



```
[Gary] 22:02:51 gwiyeong@AL01011625 ~/Documents/Project/pangyo/amicus/src/main/resources/static (master *+ u= origin/master)$ npm install --save webpack

> fsevents@1.1.2 install /Users/gwiyeong/Documents/Project/pangyo/amicus/src/main/resources/static/node_modules/fsevents
> node install

[fsevents] Success: "/Users/gwiyeong/Documents/Project/pangyo/amicus/src/main/resources/static/node_modules/fsevents/lib/binding/Release/node-v57-darwin-x
Pass --update-binary to reinstall or --build-from-source to recompile

> uglifyjs-webpack-plugin@0.4.6 postinstall /Users/gwiyeong/Documents/Project/pangyo/amicus/src/main/resources/static/node_modules/uglifyjs-webpack-plugin
> node lib/post_install.js

npm WARN amicus@1.0.0 No description
npm WARN amicus@1.0.0 No repository field.

+ webpack@3.5.5
added 362 packages in 35.279s
```

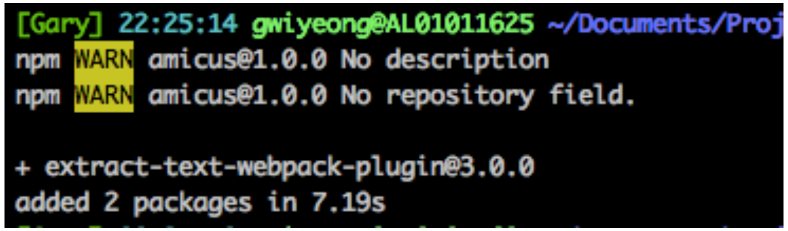
React 설치 – babel 설치

npm install --save babel-loader babel-core babel-preset-es2015 babel-preset-react babel-plugin-transform-class-properties



```
[Gary] 22:03:42 gwiyeong@AL01011625 ~/Documents/Project/pangyo/amicu
act
npm WARN amicus@1.0.0 No description
npm WARN amicus@1.0.0 No repository field.

+ babel-preset-es2015@6.24.1
+ babel-preset-react@6.24.1
+ babel-loader@7.1.2
+ babel-core@6.26.0
added 92 packages in 16.369s
```

npm install --save extract-text-webpack-plugin style-loader css-loader #text plugin for css



```
[Gary] 22:25:14 gwiyeong@AL01011625 ~/Documents/Proj
npm WARN amicus@1.0.0 No description
npm WARN amicus@1.0.0 No repository field.

+ extract-text-webpack-plugin@3.0.0
added 2 packages in 7.19s
```

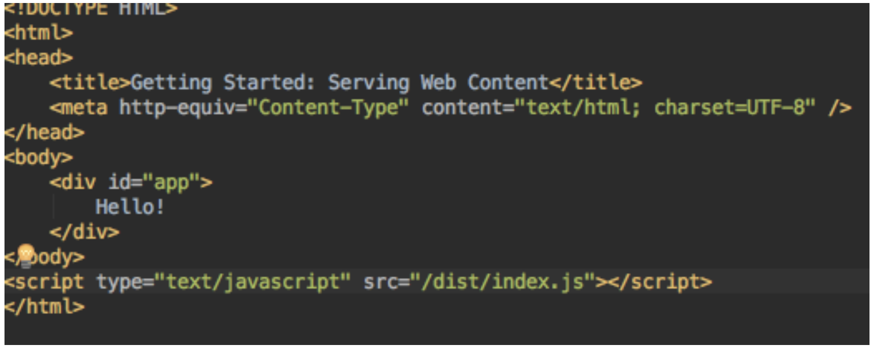
React 설치 – webpack.config.js 설정

entry : bundle의 시작점
output: bundle의 결과물

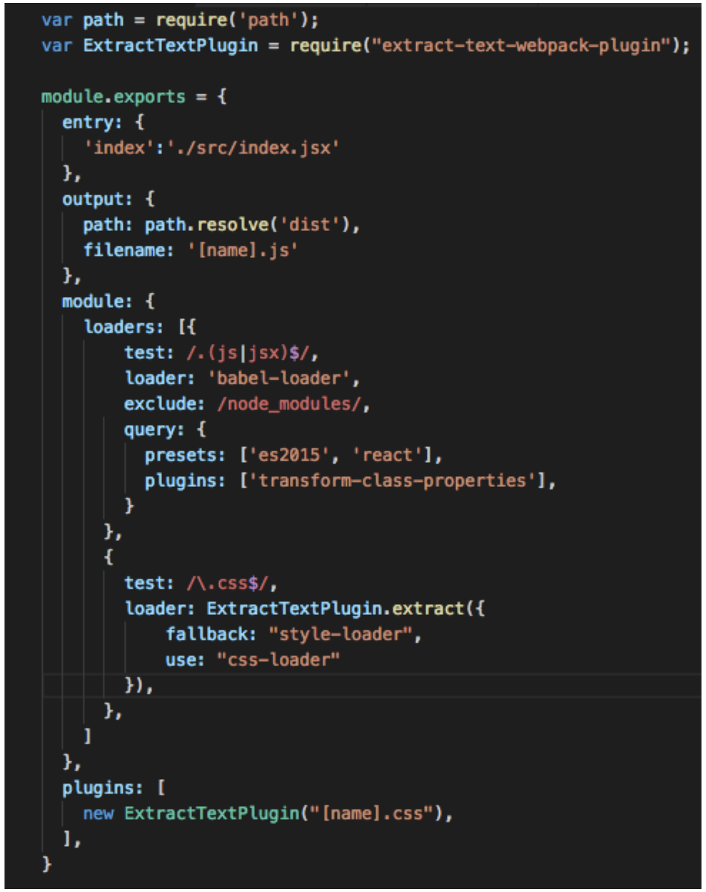
module : bundle에 사용되는 모듈

Webpack으로 build를 할 경우
./src/index.js파일을 읽어
./dist/index.js파일을 생성한다.

View에서는 ./dist/index.js를 import하도록 설정.



```html
<!DOCTYPE HTML>
<html>
<head>
    <title>Getting Started: Serving Web Content</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <div id="app">
        Hello!
    </div>
</body>
<script type="text/javascript" src="/dist/index.js"></script>
</html>
```

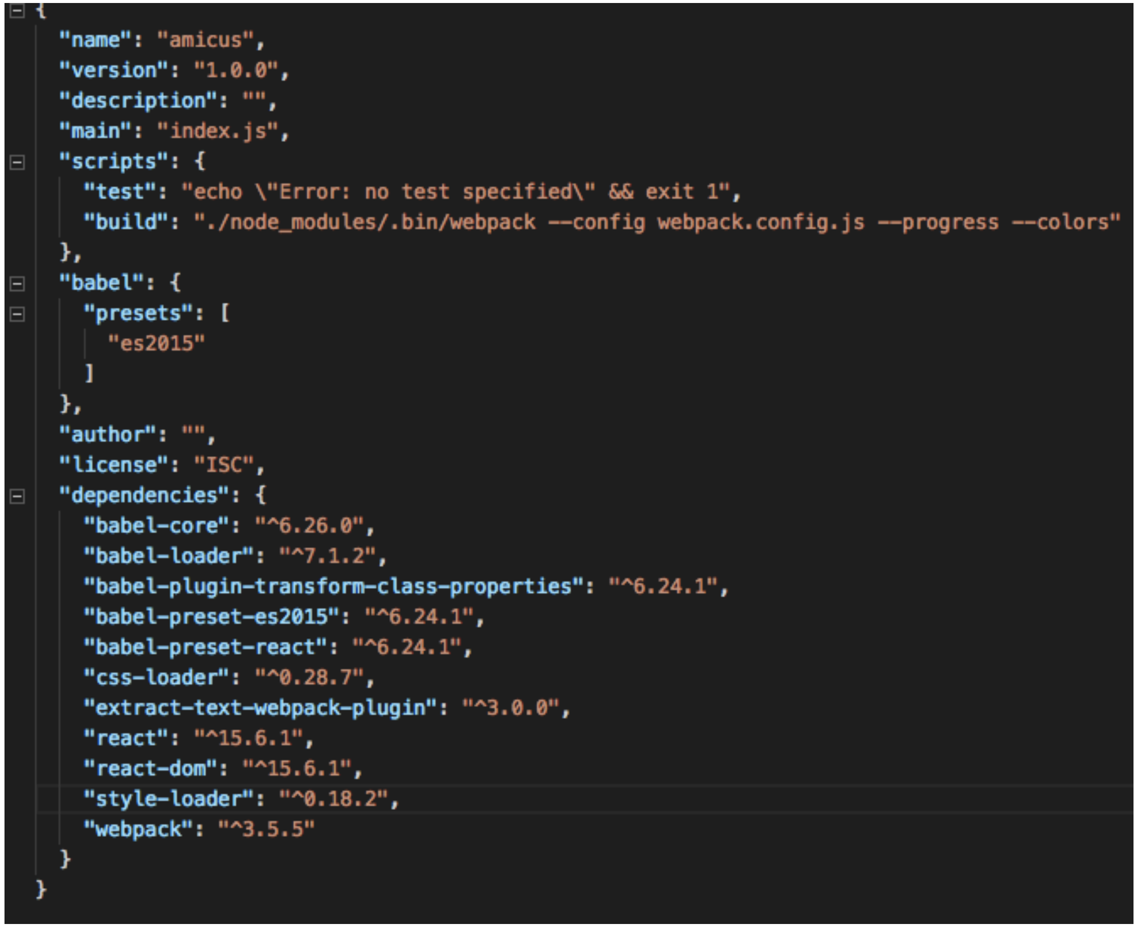


```javascript
var path = require('path');
var ExtractTextPlugin = require("extract-text-webpack-plugin");

module.exports = {
  entry: {
    'index':'./src/index.jsx'
  },
  output: {
    path: path.resolve('dist'),
    filename: '[name].js'
  },
  module: {
    loaders: [{
        test: /.(js|jsx)$/,
        loader: 'babel-loader',
        exclude: /node_modules/,
        query: {
          presets: ['es2015', 'react'],
          plugins: ['transform-class-properties'],
        }
      },
      {
        test: /\.css$/,
        loader: ExtractTextPlugin.extract({
            fallback: "style-loader",
            use: "css-loader"
        }),
      },
    ]
  },
  plugins: [
    new ExtractTextPlugin("[name].css"),
  ],
}
```

React 설치 – package.json 설정

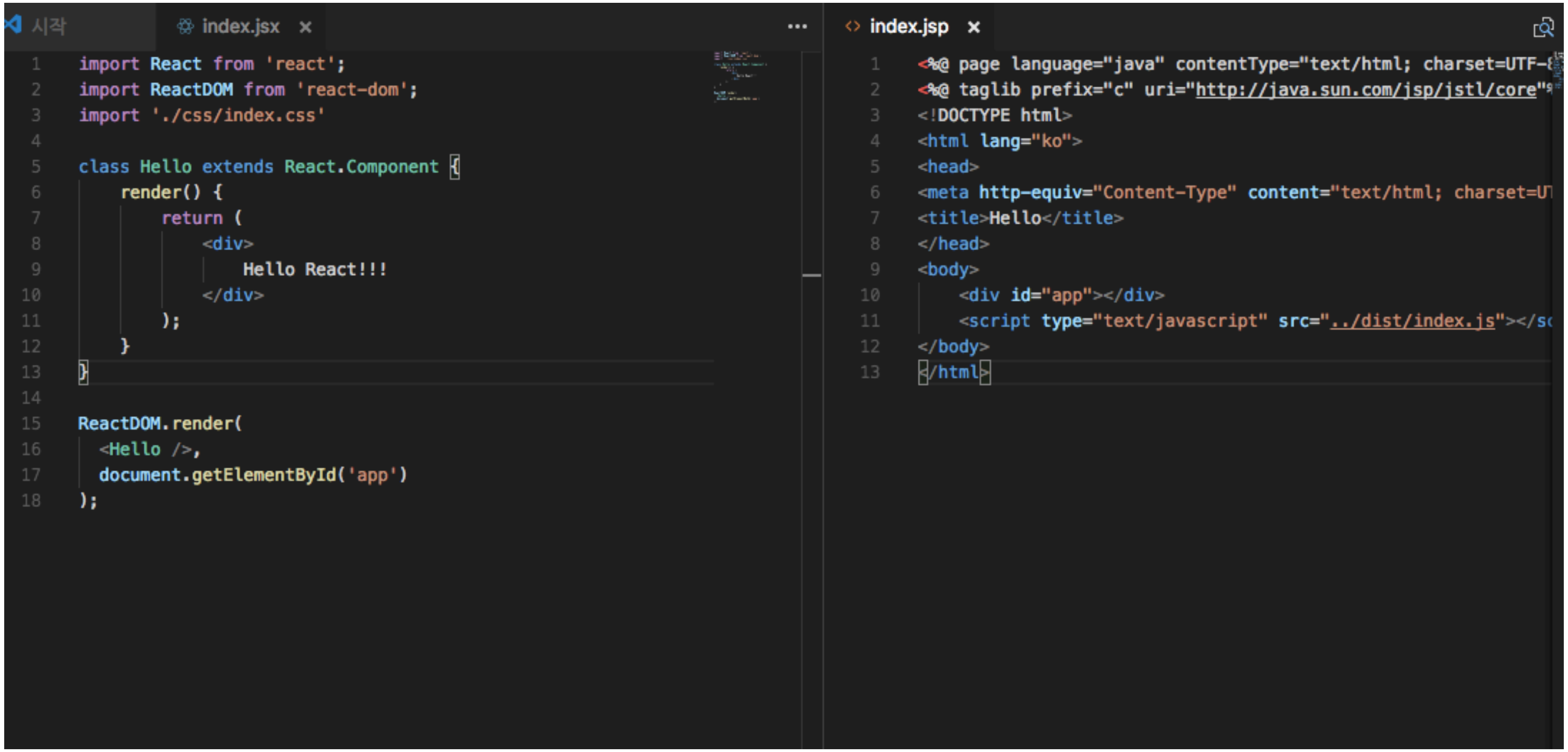
scripts.build 추가.

npm run build로 실행



```
{
  "name": "amicus",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "./node_modules/.bin/webpack --config webpack.config.js --progress --colors"
  },
  "babel": {
    "presets": [
      "es2015"
    ]
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "babel-core": "^6.26.0",
    "babel-loader": "^7.1.2",
    "babel-plugin-transform-class-properties": "^6.24.1",
    "babel-preset-es2015": "^6.24.1",
    "babel-preset-react": "^6.24.1",
    "css-loader": "^0.28.7",
    "extract-text-webpack-plugin": "^3.0.0",
    "react": "^15.6.1",
    "react-dom": "^15.6.1",
    "style-loader": "^0.18.2",
    "webpack": "^3.5.5"
  }
}
```

Hello React



```jsx
import React from 'react';
import ReactDOM from 'react-dom';
import './css/index.css'

class Hello extends React.Component {
    render() {
        return (
            <div>
                Hello React!!!
            </div>
        );
    }
}

ReactDOM.render(
  <Hello />,
  document.getElementById('app')
);
```

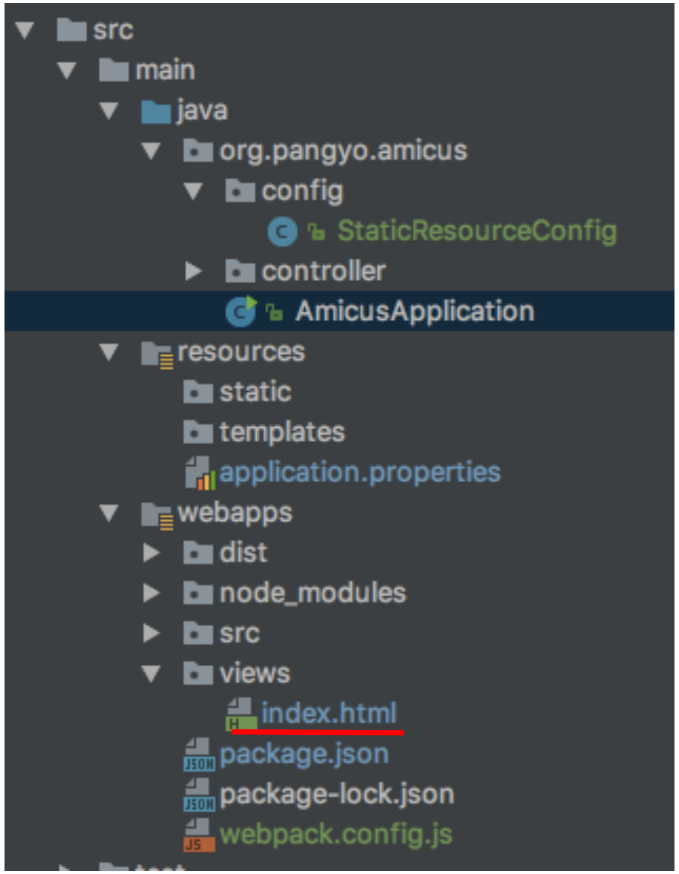
```jsp
<%@ page language="java" contentType="text/html; charset=UTF-
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"
<!DOCTYPE html>
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=U
<title>Hello</title>
</head>
<body>
    <div id="app"></div>
    <script type="text/javascript" src="../dist/index.js"></s
</body>
</html>
```

React경로와 Spring 경로 설정.

Spring이 정적 resources를 serving하는 경로와 webpack이 resources를 떨구는 경로를 맞춰야 한다!!

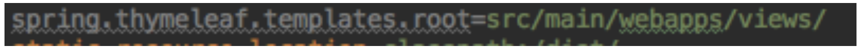
먼저 spring의 정적 resources경로를 설정한후 해당 경로 밑에 webpack이 컴파일 후 떨구도록 해야 한다.

React경로와 Spring 경로 설정.

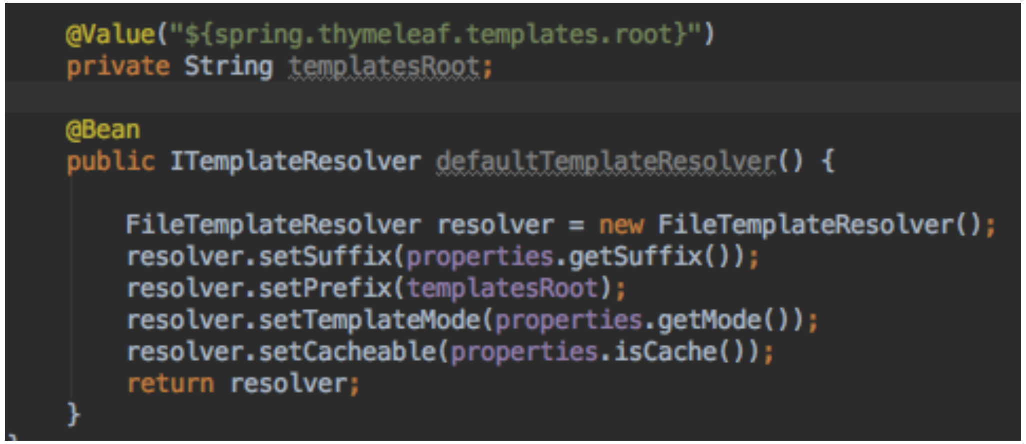


```
▼ src
  ▼ main
    ▼ java
      ▼ org.pangyo.amicus
        ▼ config
            StaticResourceConfig
        ▶ controller
          AmicusApplication
    ▼ resources
        static
        templates
        application.properties
    ▼ webapps
      ▶ dist
      ▶ node_modules
      ▶ src
      ▼ views
          index.html
        package.json
        package-lock.json
        webpack.config.js
```

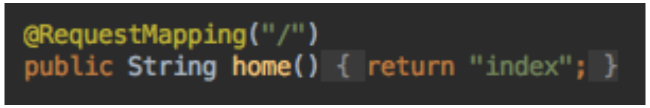
thymeleaf template의 root 경로를 설정.



```
spring.thymeleaf.templates.root=src/main/webapps/views/
```



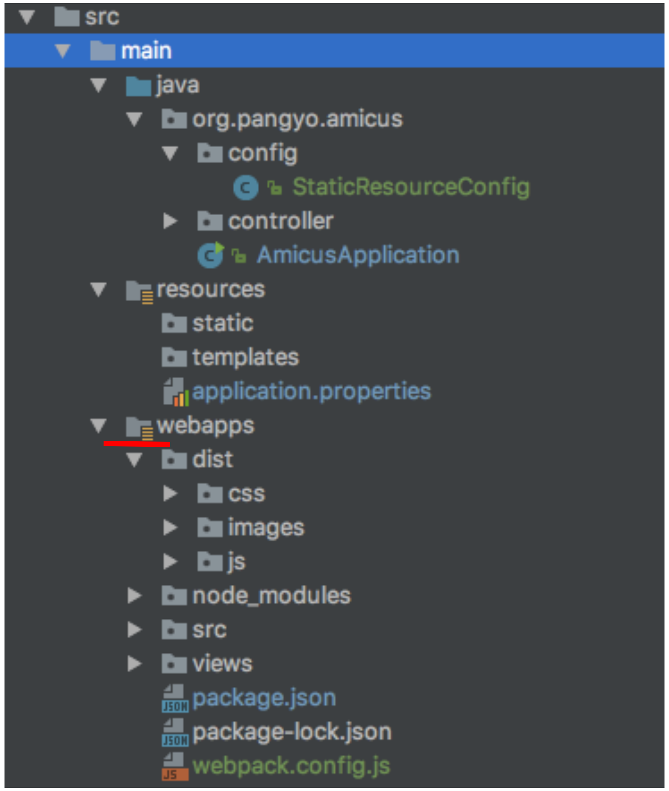
```java
    @Value("${spring.thymeleaf.templates.root}")
    private String templatesRoot;

    @Bean
    public ITemplateResolver defaultTemplateResolver() {

        FileTemplateResolver resolver = new FileTemplateResolver();
        resolver.setSuffix(properties.getSuffix());
        resolver.setPrefix(templatesRoot);
        resolver.setTemplateMode(properties.getMode());
        resolver.setCacheable(properties.isCache());
        return resolver;
    }
```



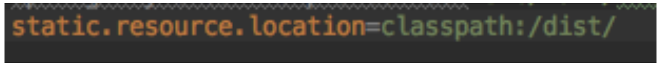
```java
@RequestMapping("/")
public String home() { return "index"; }
```

Controller에서 string return시 template를 찾아서 rendering.

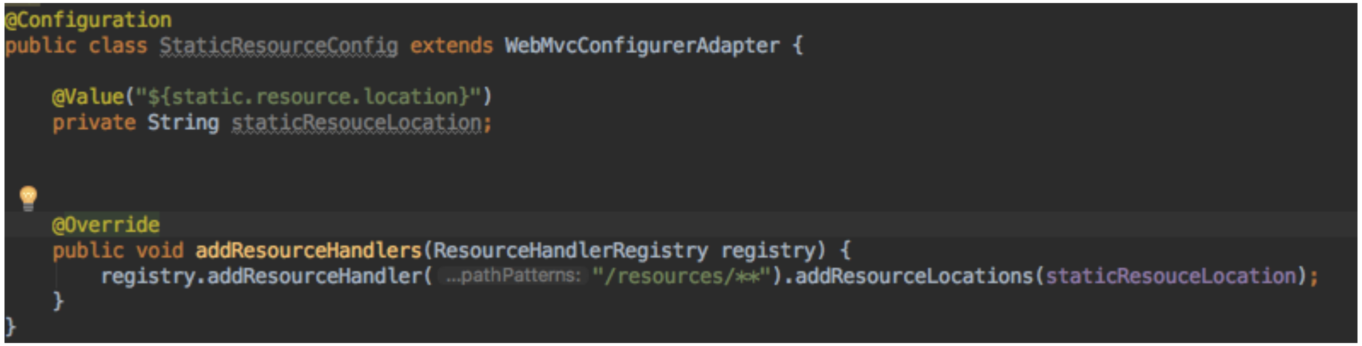
React경로와 Spring 경로 설정.



```
▼ src
  ▼ main
    ▼ java
      ▼ org.pangyo.amicus
        ▼ config
            StaticResourceConfig
        ▶ controller
          AmicusApplication
    ▼ resources
        static
        templates
        application.properties
    ▼ webapps
      ▼ dist
        ▶ css
        ▶ images
        ▶ js
      ▶ node_modules
      ▶ src
      ▶ views
        package.json
        package-lock.json
        webpack.config.js
```



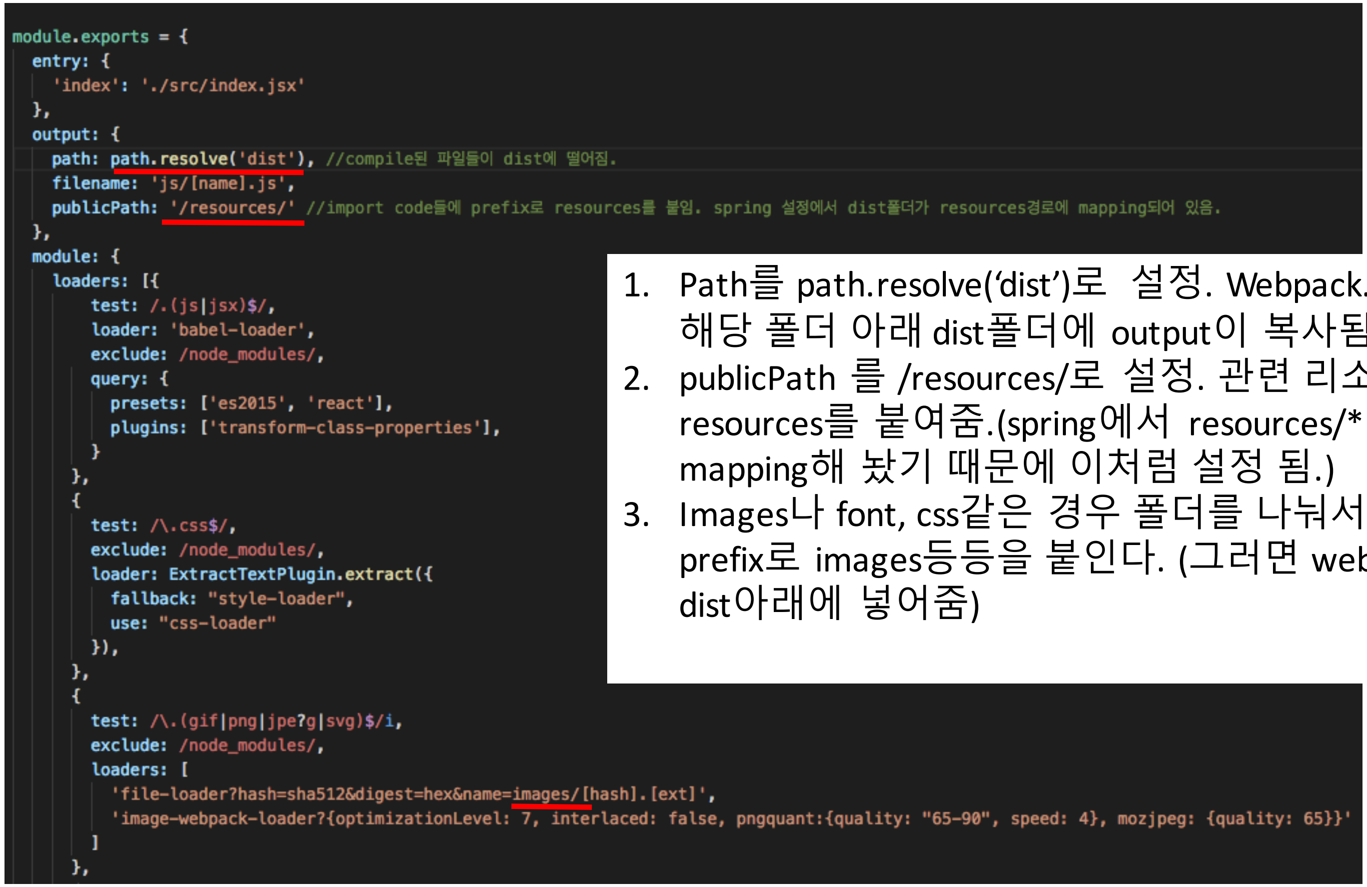
```
static.resource.location=classpath:/dist/
```



```java
@Configuration
public class StaticResourceConfig extends WebMvcConfigurerAdapter {

    @Value("${static.resource.location}")
    private String staticResouceLocation;


    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler( ...pathPatterns: "/resources/**").addResourceLocations(staticResouceLocation);
    }
}
```

1. webapps를 resources root로 설정 (build시 target.class폴더 밑에 webapps의 모든 파일/폴더 들이 복사된다.)
2. static.resource.location을 classpath:/dist/로 설정. (target.class폴더가 classpath로 잡혀있음)
3. /resources/경로에 classpath:/dist/경로를 mapping.

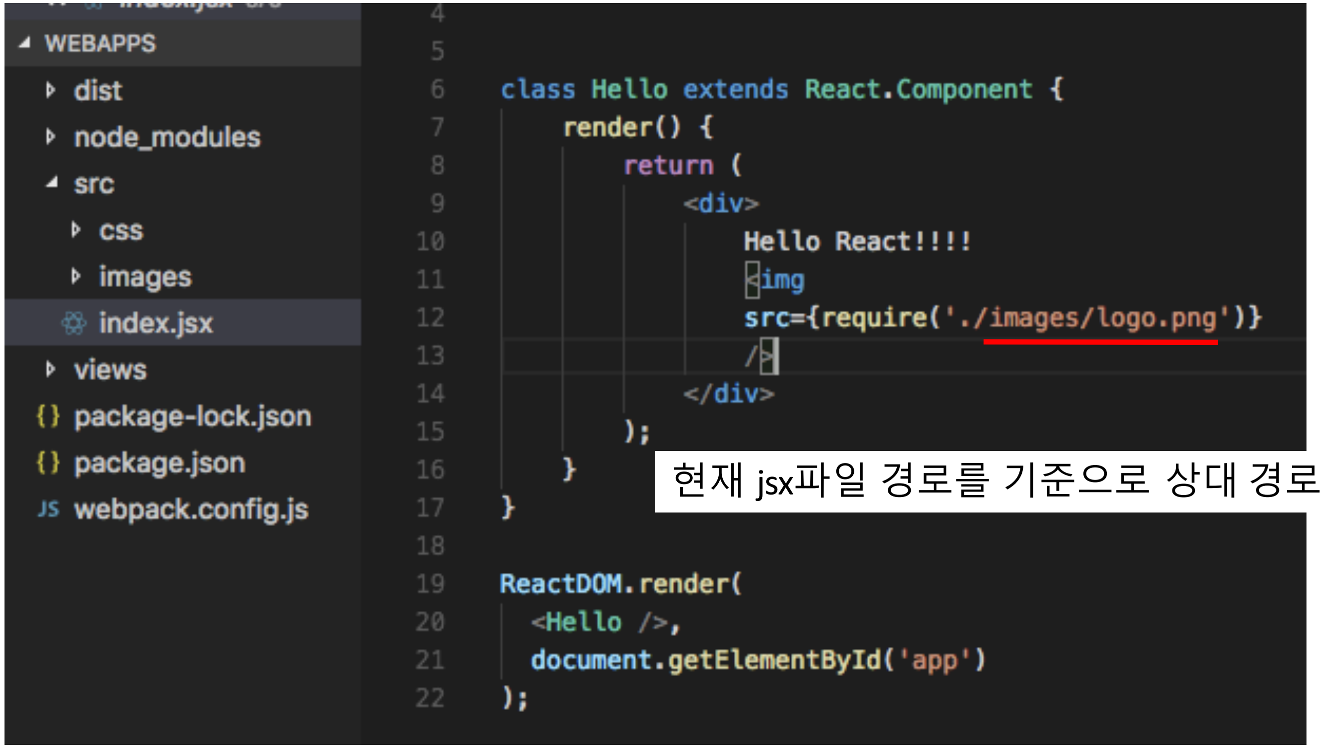
React경로와 Spring 경로 설정.



```
module.exports = {
  entry: {
    'index': './src/index.jsx'
  },
  output: {
    path: path.resolve('dist'), //compile된 파일들이 dist에 떨어짐.
    filename: 'js/[name].js',
    publicPath: '/resources/' //import code들에 prefix로 resources를 붙임. spring 설정에서 dist폴더가 resources경로에 mapping되어 있음.
  },
  module: {
    loaders: [{
        test: /.(js|jsx)$/,
        loader: 'babel-loader',
        exclude: /node_modules/,
        query: {
          presets: ['es2015', 'react'],
          plugins: ['transform-class-properties'],
        }
      },
      {
        test: /\.css$/,
        exclude: /node_modules/,
        loader: ExtractTextPlugin.extract({
          fallback: "style-loader",
          use: "css-loader"
        }),
      },
      {
        test: /\.(gif|png|jpe?g|svg)$/i,
        exclude: /node_modules/,
        loaders: [
          'file-loader?hash=sha512&digest=hex&name=images/[hash].[ext]',
          'image-webpack-loader?{optimizationLevel: 7, interlaced: false, pngquant:{quality: "65-90", speed: 4}, mozjpeg: {quality: 65}}'
        ]
      },
```

1. Path를 path.resolve('dist')로 설정. Webpack.config가 있는 폴더를 기준으로 해당 폴더 아래 dist폴더에 output이 복사됨.
2. publicPath 를 /resources/로 설정. 관련 리소스를 가져오는 부분에서 prefix로 resources를 붙여줌.(spring에서 resources/**에 대해서 Static resource를 mapping해 놨기 때문에 이처럼 설정 됨.)
3. Images나 font, css같은 경우 폴더를 나눠서 load하고 싶으면 loader에 prefix로 images등등을 붙인다. (그러면 webpack이 알아서 폴더 나눠서 dist아래에 넣어줌)

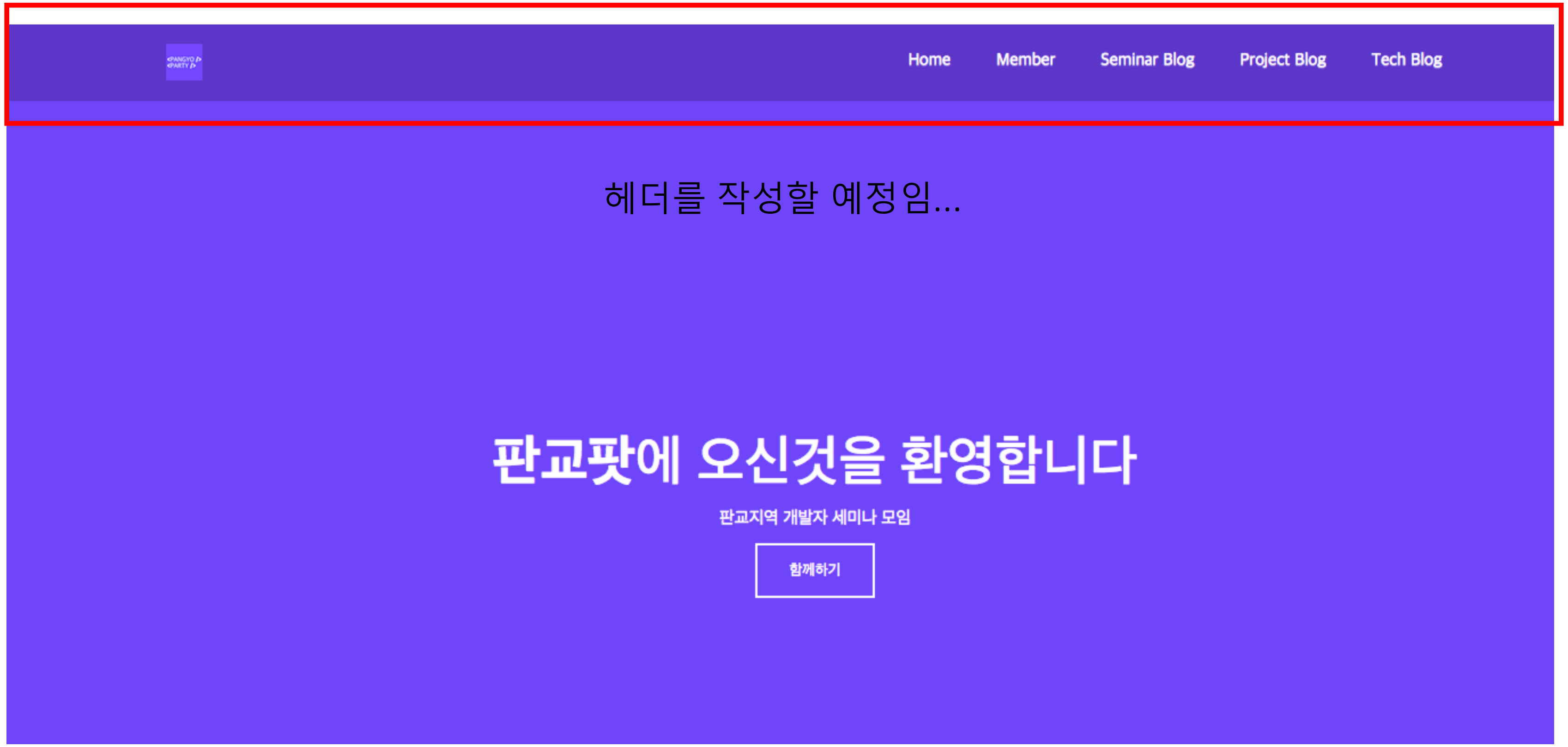
React경로와 Spring 경로 설정.



```
class Hello extends React.Component {
    render() {
        return (
            <div>
                Hello React!!!!
                <img
                src={require('./images/logo.png')}
                />
            </div>
        );
    }
}

ReactDOM.render(
  <Hello />,
  document.getElementById('app')
);
```

현재 jsx파일 경로를 기준으로 상대 경로 image file을 위와 같이 입력하면 load됨

React Component



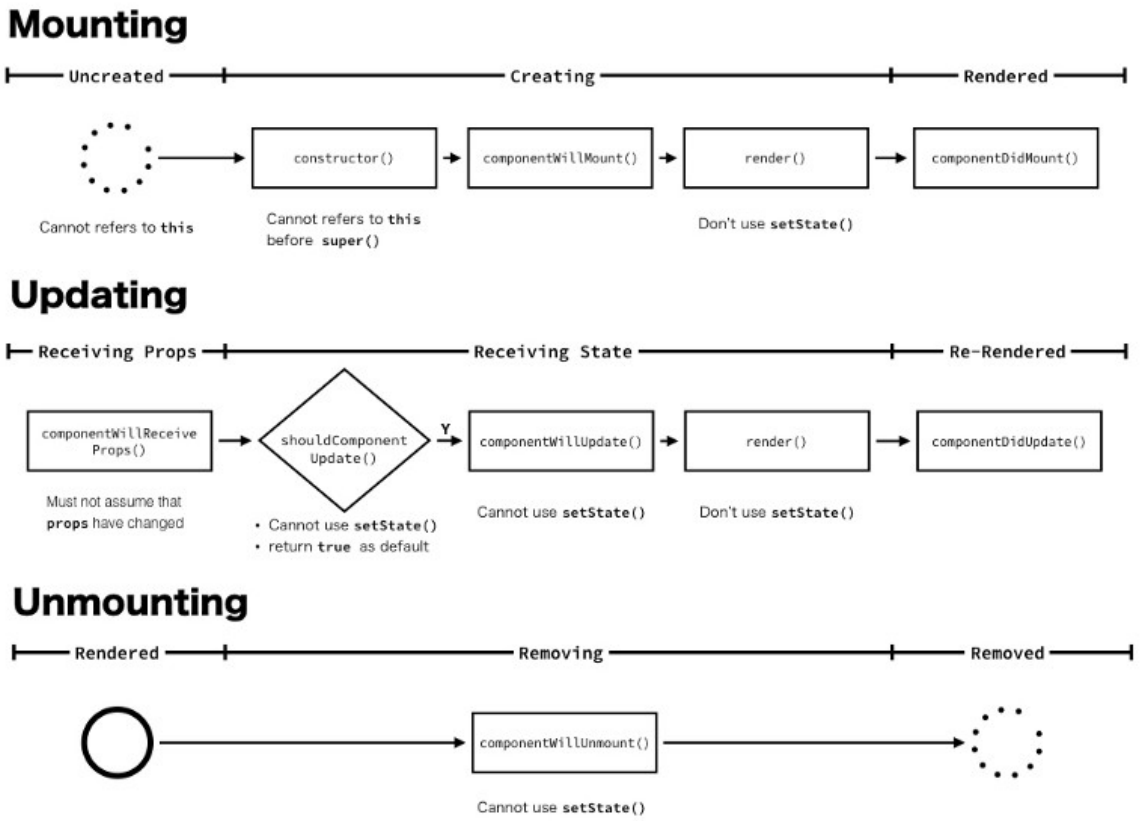
헤더를 작성할 예정임...

판교팟에 오신것을 환영합니다

판교지역 개발자 세미나 모임

React Component

Component Life Cycle



Mounting

Uncreated — Creating — Rendered

constructor() → componentWillMount() → render() → componentDidMount()

Cannot refers to **this**

Cannot refers to **this** before **super()**

Don't use **setState()**

Updating

Receiving Props — Receiving State — Re-Rendered

componentWillReceiveProps() → shouldComponentUpdate() →Y componentWillUpdate() → render() → componentDidUpdate()

Must not assume that **props** have changed

- Cannot use **setState()**
- return **true** as default

Cannot use **setState()**

Don't use **setState()**

Unmounting

Rendered — Removing — Removed

componentWillUnmount()

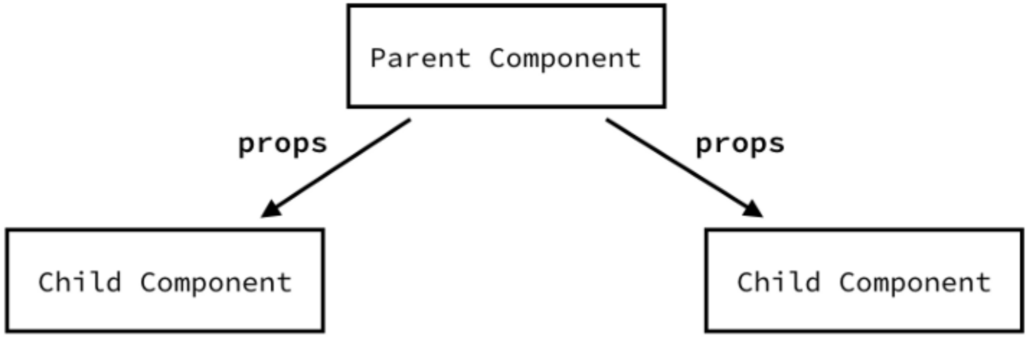
Cannot use **setState()**

React Component

Props, State

Props : 자식 컴포넌트 전달 하는 데이터
단방향 전달(unidirection).

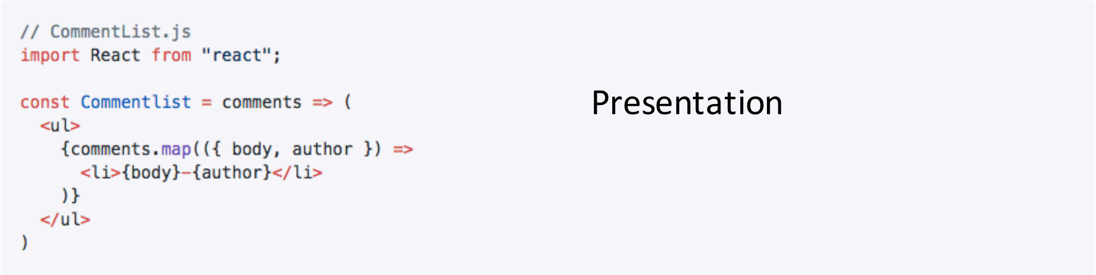
State : 현재 컴포넌트가 가진 데이터.
setState로 state를 update한다.
setState – 비동기적으로 동작하고,
새로운 오브젝트를 생성하여 state를
변형시키기 때문에 reference값이 변경되어
참조하는 모든 컴포넌트가 변경을 인지할수 있고, rerender 된다.
callback은 setState(nextState, callback)으로 작업 해야함.



React Component

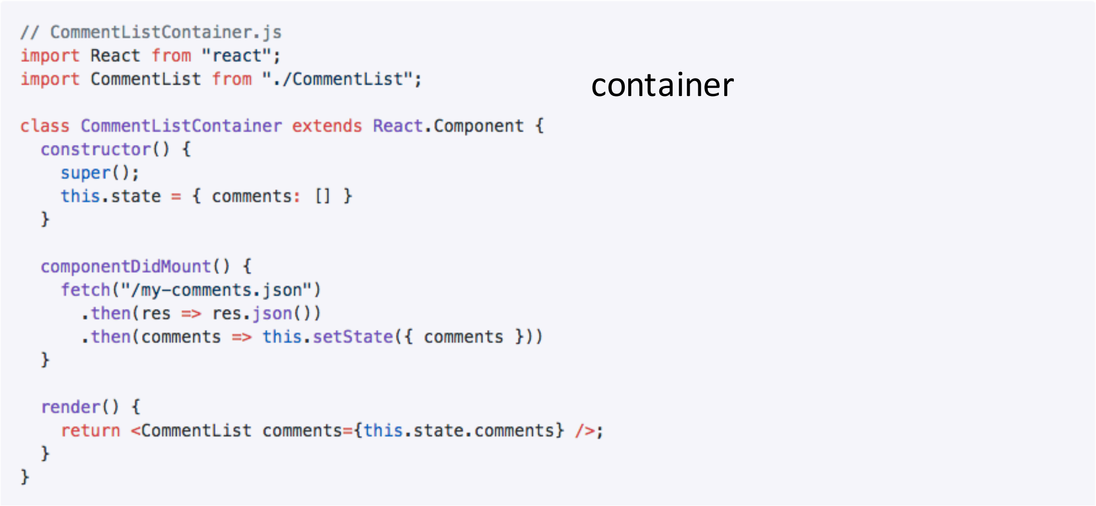
Presentation Component : render에 필요한 데이터만 입력하면 view를 그려주는 component

Container Component : 마크업 보다 비즈니스 로직에 집중된, ajax, HOC(High Order Component)등을 이용해 데이터를 fetching하는 component.



```
// CommentList.js
import React from "react";

const Commentlist = comments => (
  <ul>
    {comments.map(({ body, author }) =>
      <li>{body}-{author}</li>
    )}
  </ul>
)
```

Presentation



```
// CommentListContainer.js
import React from "react";
import CommentList from "./CommentList";

class CommentListContainer extends React.Component {
  constructor() {
    super();
    this.state = { comments: [] }
  }

  componentDidMount() {
    fetch("/my-comments.json")
      .then(res => res.json())
      .then(comments => this.setState({ comments }))
  }

  render() {
    return <CommentList comments={this.state.comments} />;
  }
}
```

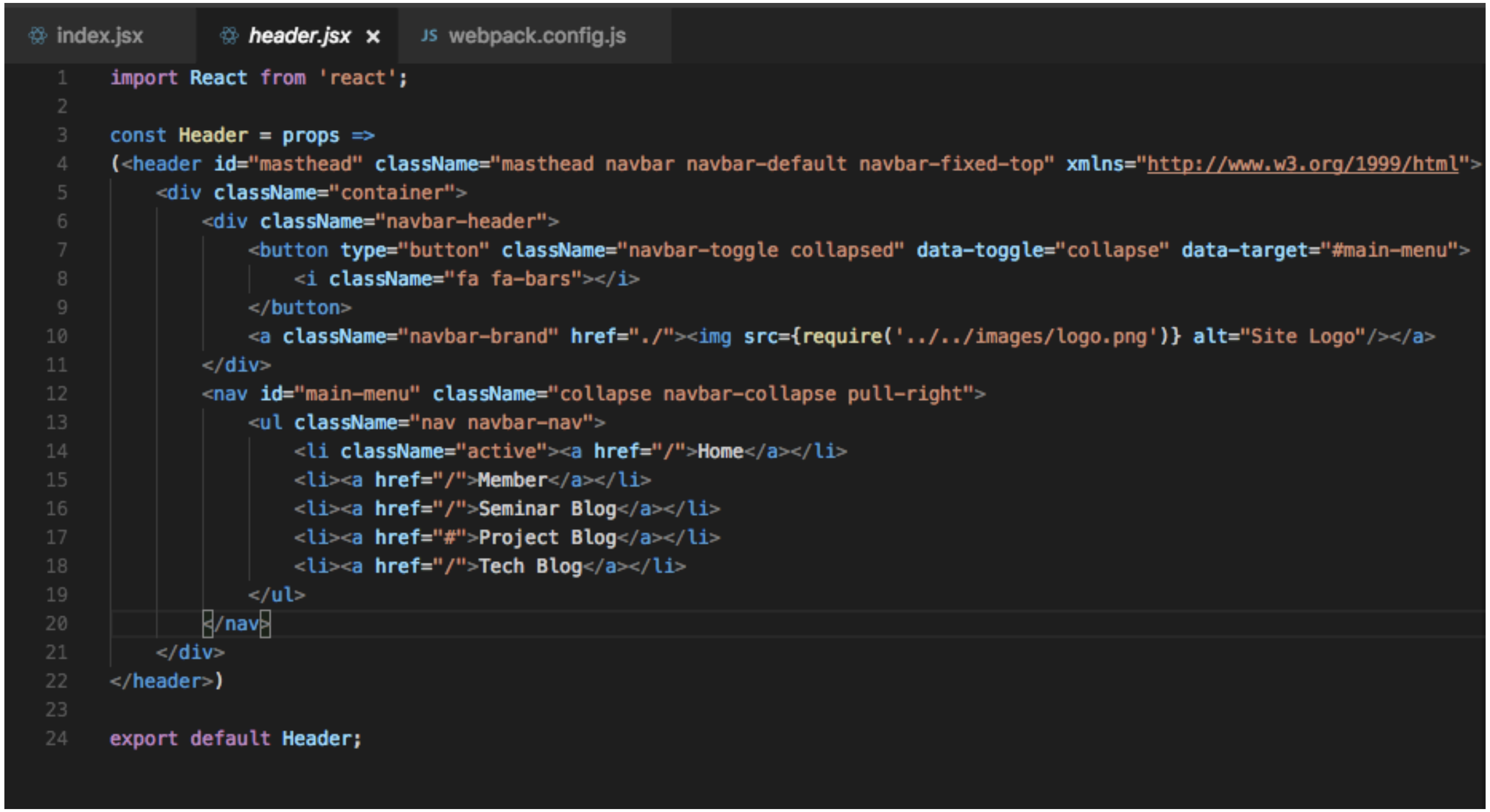
container

React Component

Header

Presentation component
로 구현

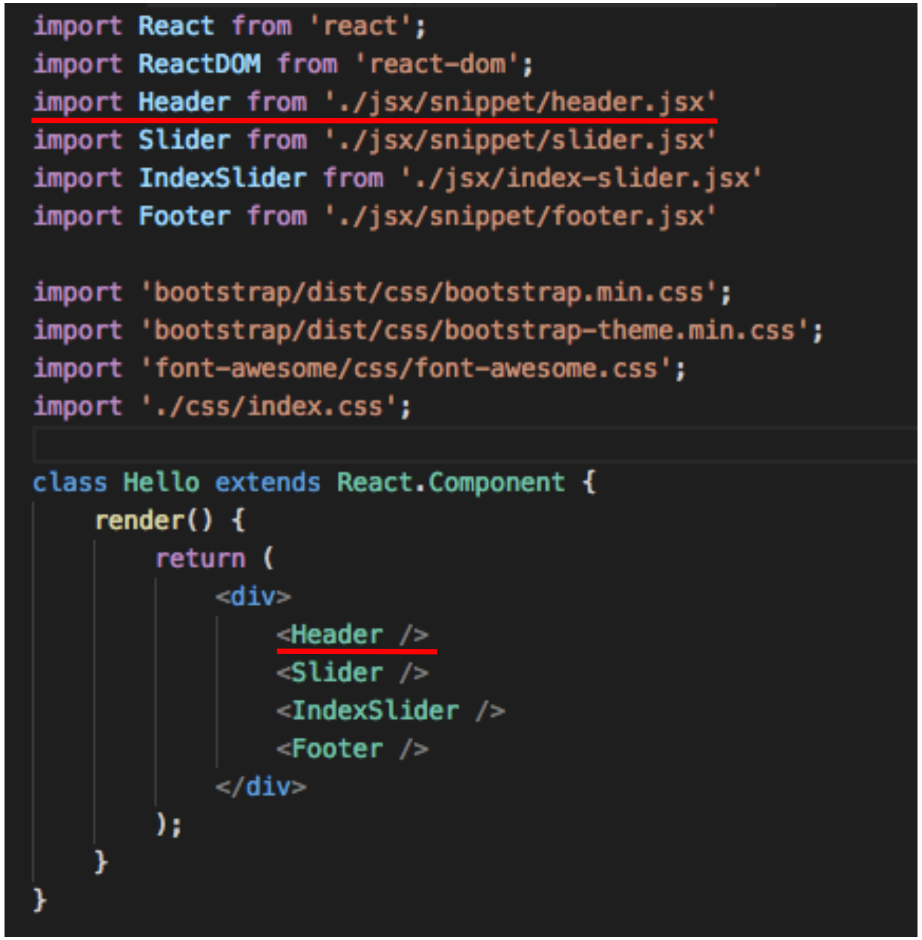
Img경로를 require js로
입력



```jsx
import React from 'react';

const Header = props =>
(<header id="masthead" className="masthead navbar navbar-default navbar-fixed-top" xmlns="http://www.w3.org/1999/html">
    <div className="container">
        <div className="navbar-header">
            <button type="button" className="navbar-toggle collapsed" data-toggle="collapse" data-target="#main-menu">
                <i className="fa fa-bars"></i>
            </button>
            <a className="navbar-brand" href="./"><img src={require('../../images/logo.png')} alt="Site Logo"/></a>
        </div>
        <nav id="main-menu" className="collapse navbar-collapse pull-right">
            <ul className="nav navbar-nav">
                <li className="active"><a href="/">Home</a></li>
                <li><a href="/">Member</a></li>
                <li><a href="/">Seminar Blog</a></li>
                <li><a href="#">Project Blog</a></li>
                <li><a href="/">Tech Blog</a></li>
            </ul>
        </nav>
    </div>
</header>)

export default Header;
```

React Component

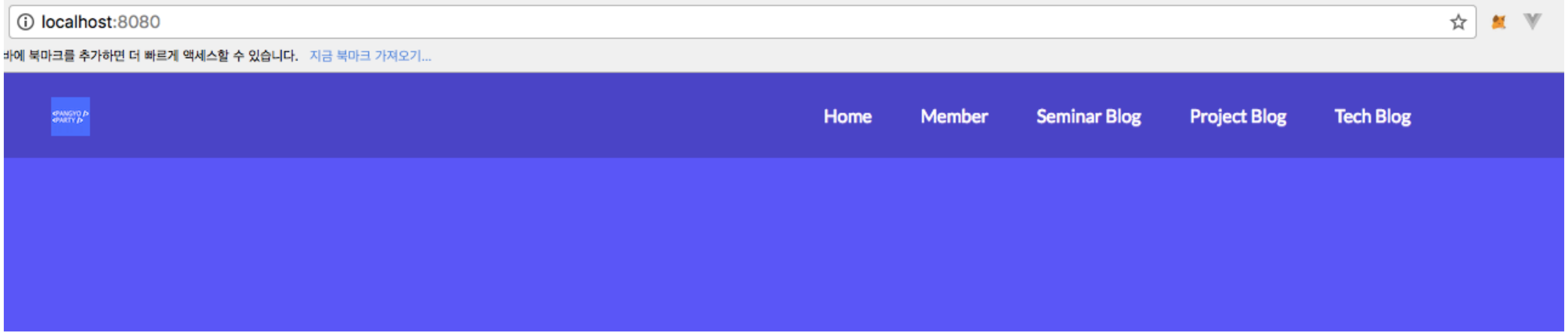
구현된 header를 import하고
Header 태그로 입력.



```
import React from 'react';
import ReactDOM from 'react-dom';
import Header from './jsx/snippet/header.jsx'
import Slider from './jsx/snippet/slider.jsx'
import IndexSlider from './jsx/index-slider.jsx'
import Footer from './jsx/snippet/footer.jsx'

import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/css/bootstrap-theme.min.css';
import 'font-awesome/css/font-awesome.css';
import './css/index.css';

class Hello extends React.Component {
    render() {
        return (
            <div>
                <Header />
                <Slider />
                <IndexSlider />
                <Footer />
            </div>
        );
    }
}
```

React Component

완성.. 하지만...



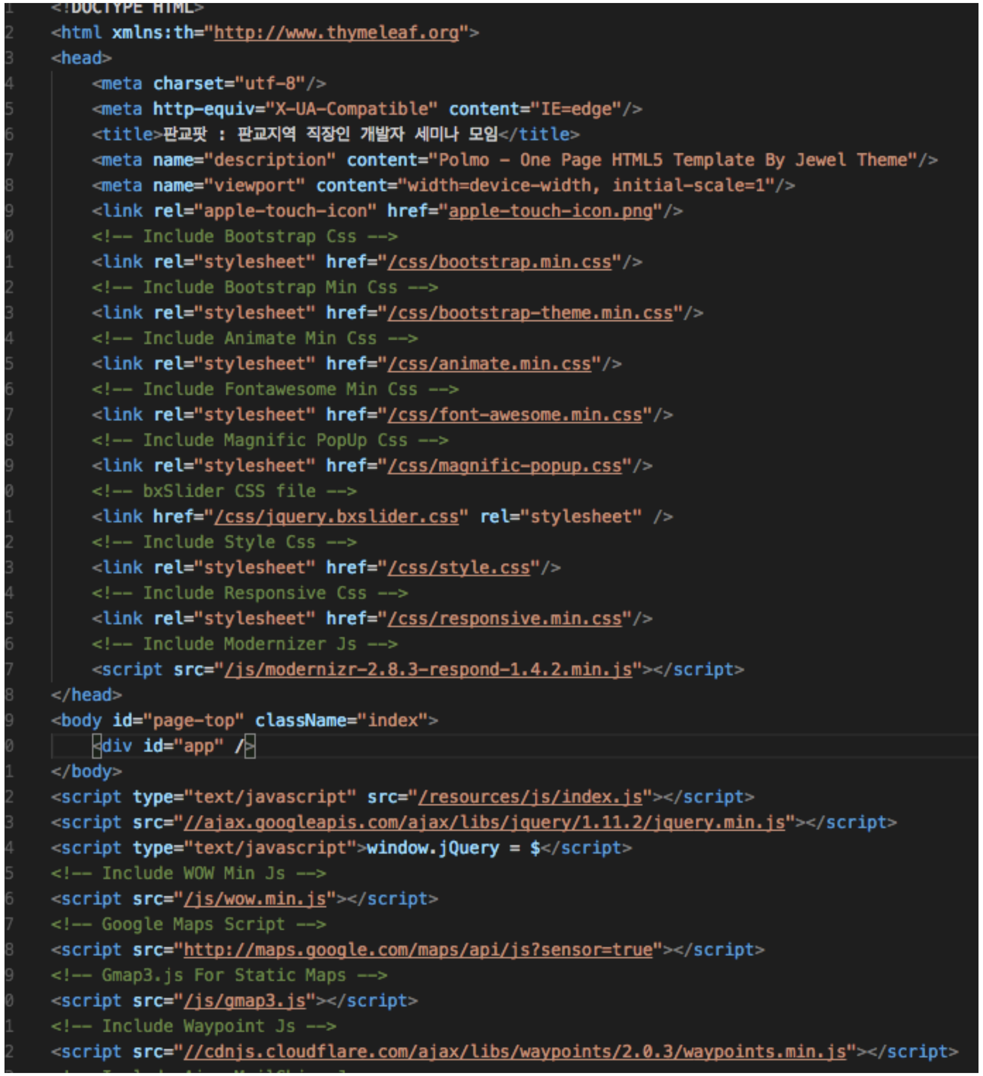
localhost:8080

바에 북마크를 추가하면 더 빠르게 액세스할 수 있습니다. 지금 북마크 가져오기...

React Component

Index.html

수많은 javascript , css dependency를 npm dependency로 변환하는 과정을 실패함...
그래서 어쩔수 없이 html에 전부다 박아넣음..
ㅠㅠ
Main 페이지에 Javascript효과들이 너무 많아서 실패... 다른 페이지 부터는 전부 새로운 html에 npm dependency로 작성할 예정.



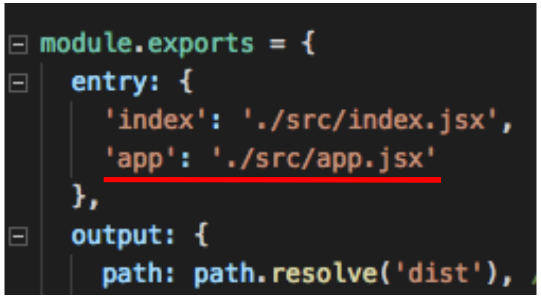
```html
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <title>판교팟 : 판교지역 직장인 개발자 세미나 모임</title>
    <meta name="description" content="Polmo - One Page HTML5 Template By Jewel Theme"/>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <link rel="apple-touch-icon" href="apple-touch-icon.png"/>
    <!-- Include Bootstrap Css -->
    <link rel="stylesheet" href="/css/bootstrap.min.css"/>
    <!-- Include Bootstrap Min Css -->
    <link rel="stylesheet" href="/css/bootstrap-theme.min.css"/>
    <!-- Include Animate Min Css -->
    <link rel="stylesheet" href="/css/animate.min.css"/>
    <!-- Include Fontawesome Min Css -->
    <link rel="stylesheet" href="/css/font-awesome.min.css"/>
    <!-- Include Magnific PopUp Css -->
    <link rel="stylesheet" href="/css/magnific-popup.css"/>
    <!-- bxSlider CSS file -->
    <link href="/css/jquery.bxslider.css" rel="stylesheet" />
    <!-- Include Style Css -->
    <link rel="stylesheet" href="/css/style.css"/>
    <!-- Include Responsive Css -->
    <link rel="stylesheet" href="/css/responsive.min.css"/>
    <!-- Include Modernizer Js -->
    <script src="/js/modernizr-2.8.3-respond-1.4.2.min.js"></script>
</head>
<body id="page-top" className="index">
    <div id="app" />
</body>
<script type="text/javascript" src="/resources/js/index.js"></script>
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script type="text/javascript">window.jQuery = $</script>
<!-- Include WOW Min Js -->
<script src="/js/wow.min.js"></script>
<!-- Google Maps Script -->
<script src="http://maps.google.com/maps/api/js?sensor=true"></script>
<!-- Gmap3.js For Static Maps -->
<script src="/js/gmap3.js"></script>
<!-- Include Waypoint Js -->
<script src="//cdnjs.cloudflare.com/ajax/libs/waypoints/2.0.3/waypoints.min.js"></script>
```

React Component

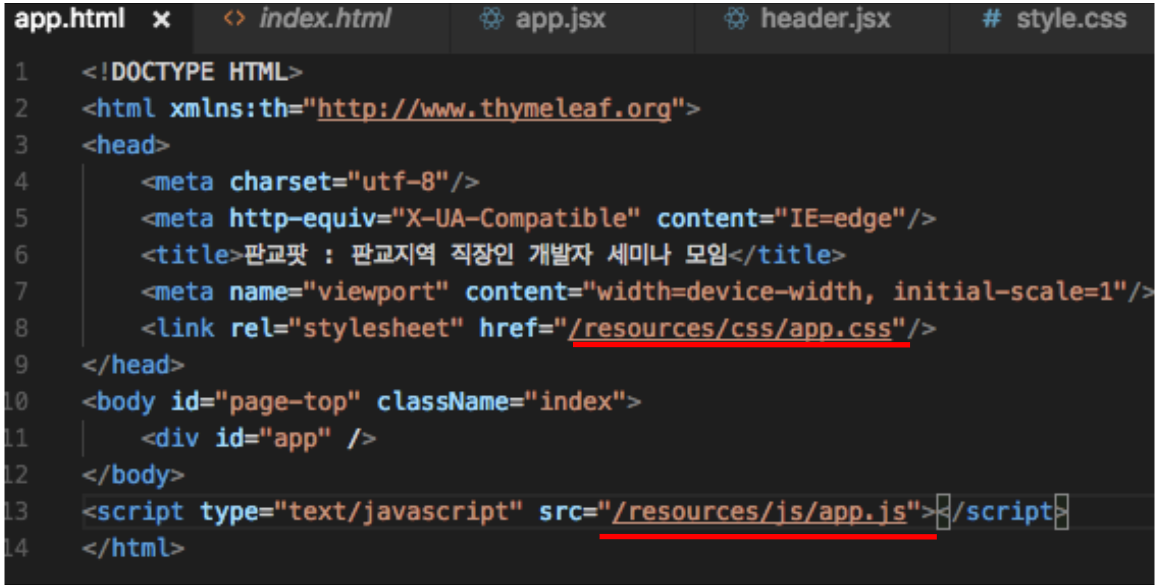
app.html

Index페이지를 제외한 나머지 페이지를 SPA로 개발하기 위해 새로운 html을 생성.

webpack.config.js 수정



```
module.exports = {
  entry: {
    'index': './src/index.jsx',
    'app': './src/app.jsx'
  },
  output: {
    path: path.resolve('dist'),
```

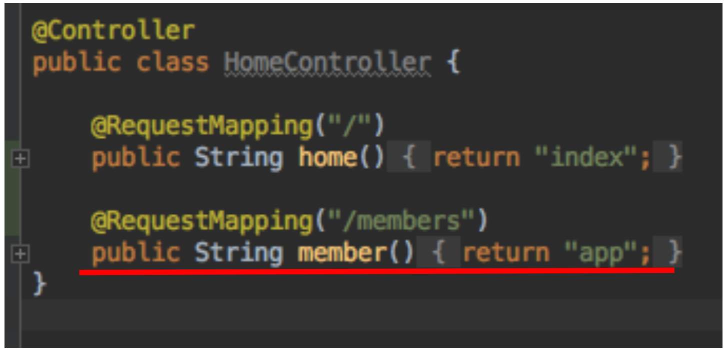


```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <title>판교팟 : 판교지역 직장인 개발자 세미나 모임</title>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <link rel="stylesheet" href="/resources/css/app.css"/>
</head>
<body id="page-top" className="index">
    <div id="app" />
</body>
<script type="text/javascript" src="/resources/js/app.js"></script>
</html>
```

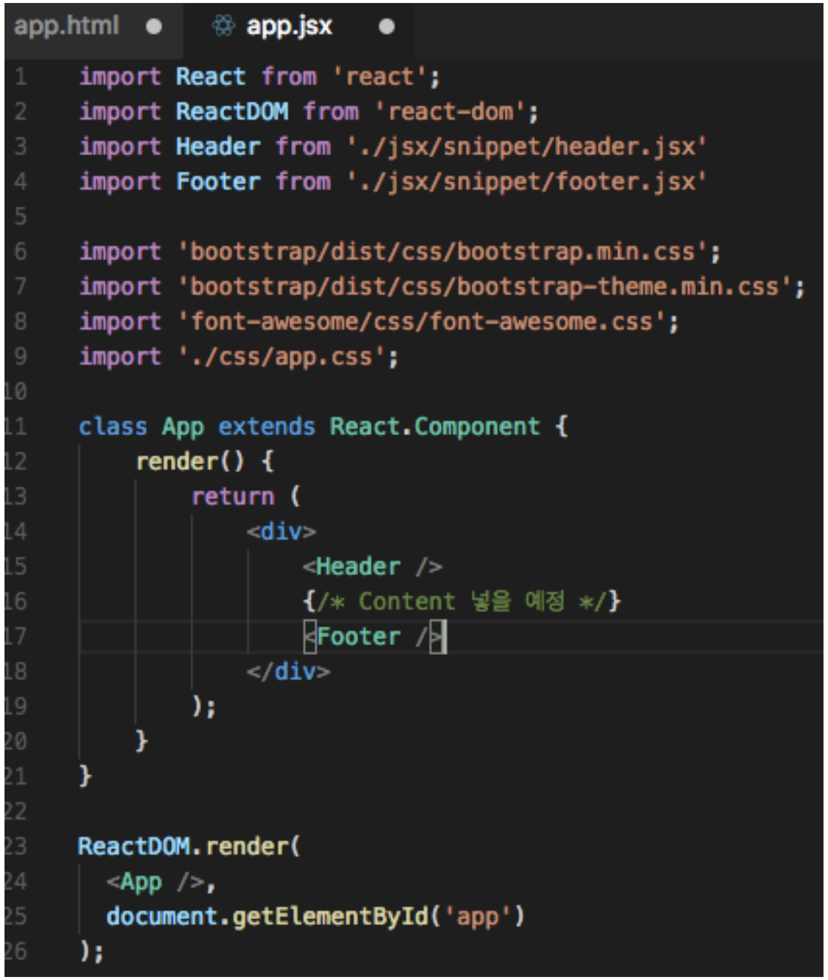
React Component

app.jsx

Index페이지를 제외한 나머지 페이지를 SPA로
개발하기 위해 새로운 jsx을 생성.
기본으로 bootstrap과 fontawesome을 넣어 봄.



```java
@Controller
public class HomeController {

    @RequestMapping("/")
    public String home() { return "index"; }

    @RequestMapping("/members")
    public String member() { return "app"; }
}
```

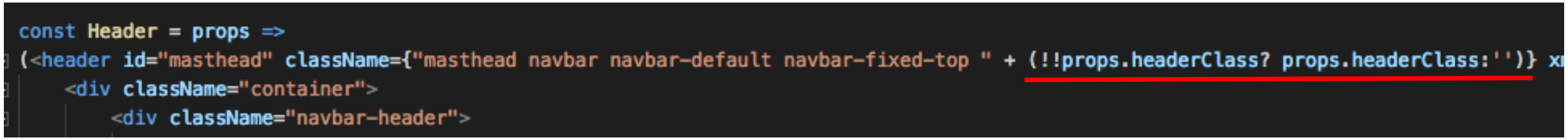


```jsx
import React from 'react';
import ReactDOM from 'react-dom';
import Header from './jsx/snippet/header.jsx'
import Footer from './jsx/snippet/footer.jsx'

import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/css/bootstrap-theme.min.css';
import 'font-awesome/css/font-awesome.css';
import './css/app.css';

class App extends React.Component {
    render() {
        return (
            <div>
                <Header />
                {/* Content 넣을 예정 */}
                <Footer />
            </div>
        );
    }
}

ReactDOM.render(
  <App />,
  document.getElementById('app')
);
```

React Component

header.jsx

Header를 index page와 app page에서 동일한 컴포넌드로 활용하기 위해 추가적인 classname을 parent에서 받아서 dynamic하게 classname을 추가할수 있도록 변경함.



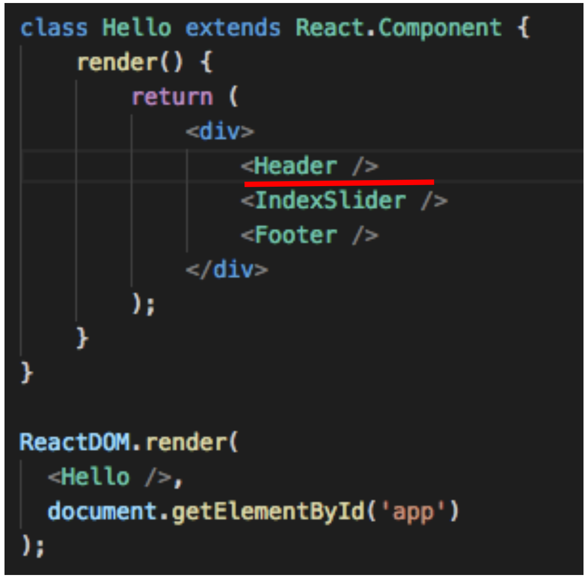
```
const Header = props =>
(<header id="masthead" className={"masthead navbar navbar-default navbar-fixed-top " + (!!props.headerClass? props.headerClass:'')} xr
    <div className="container">
        <div className="navbar-header">
```

<Header headerClass='test' /> 라고 child를 선언하게 되면 props에 {directive key : value}의 객체로 들어오게 된다.

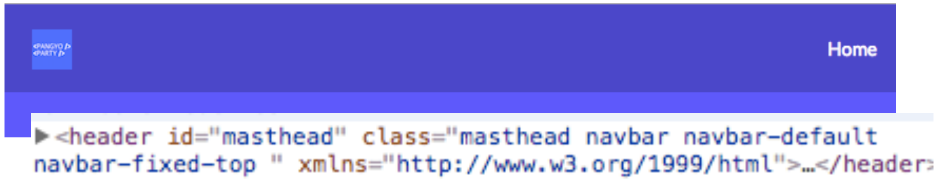
className={javascript 로직}로 dynamic하게 추가.

React Component

index.jsx

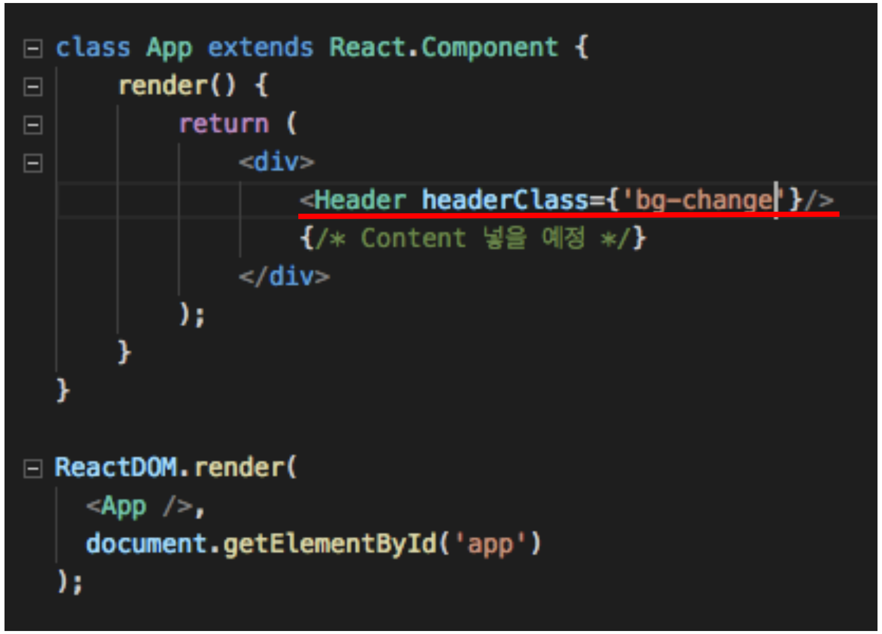


```
class Hello extends React.Component {
    render() {
        return (
            <div>
                <Header />
                <IndexSlider />
                <Footer />
            </div>
        );
    }
}

ReactDOM.render(
  <Hello />,
  document.getElementById('app')
);
```

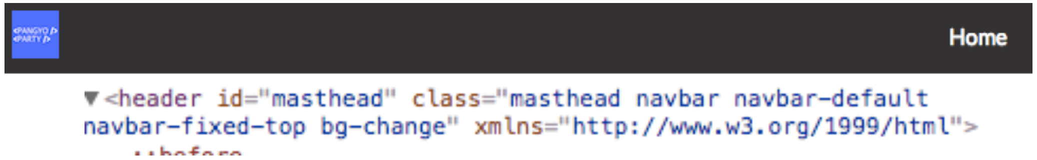


```
▶<header id="masthead" class="masthead navbar navbar-default
navbar-fixed-top " xmlns="http://www.w3.org/1999/html">…</header>
```

app.jsx



```
class App extends React.Component {
    render() {
        return (
            <div>
                <Header headerClass={'bg-change'}/>
                {/* Content 넣을 예정 */}
            </div>
        );
    }
}

ReactDOM.render(
    <App />,
    document.getElementById('app')
);
```



```
▼<header id="masthead" class="masthead navbar navbar-default
navbar-fixed-top bg-change" xmlns="http://www.w3.org/1999/html">
  ::before
```