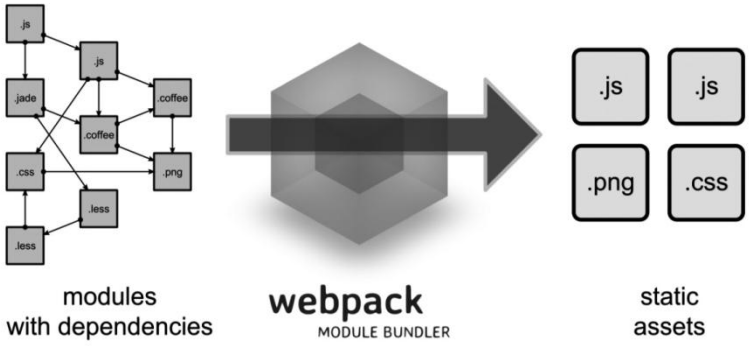
Webpack

Webpack?

- 자바스크립트 애플리케이션 개발을 위한 모듈 번들러
- 모듈, 정적 자원들을 묶어 하나 또는 몇개의 파일로 번들링함
- 장점
  - 초기로딩 타임 감소
  - 정적 자원들까지 모듈화 가능
  - 트랜스파일러들과 쉽게 통합
  - HMR(Hot Module Replacement)를 지원 -> 코드 수정시 자동으로 번들링, 페이지 갱신
  - 다양한 로더(Loader), 플러그인(Plugin) 지원



Webpack 구성

Webpack 구성 파일

- webpack 실행시 반드시 필요
- 파일명을 지정하지 않으면 webpack.config.js 가 사용됨
- 핵심 옵션 객체
  - entry
  - output
  - module
  - plugins

```
var webpack = require('webpack');
module.exports = {
    entry: {

    },
    output: {

    },
    module: {
        rules: [

        ]
    },
    plugins: [

    ],
    ……
}
```

Webpack 구성

entry 옵션

- 처음 로드하는 파일 지정

```
[ 단일 진입 파일인 경우 ]
entry: __dirname + '/src/index.js'

[ 다중 진입 파일인 경우 ]
entry: {
  main: __dirname + '/src/index.js',
  app: __dirname + '/src/main.js'
}
```

output

- 번들링된 결과물의 출력방법을 지정함.
- [name] 대체 문자 : entry 옵션의 진입 파일의 이름을 출력으로...

```
output: {
    path: __dirname + '/public/dist/'),
    filename: '[name].js',
    publicPath : '/dist'
}
```

Webpack 구성

module

- 프로젝트 내부에서 사용하는 다양한 모듈의 수행방법 지정
- 특히 loader!!

```
module: {
      rules: [
         {
            test:/₩.js$/,
            loader: 'babel-loader',
            exclude: /node_modules/
         }
      ]
  },
```

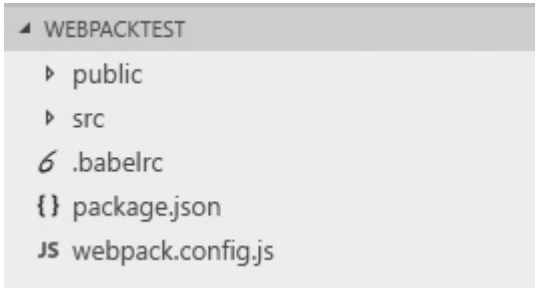
plugins

- webpack 빌드 프로세스에 사용자가 지정한 작업을 추가할 수 있는 기능

```
plugins : [
   new webpack.optimize.UglifyJsPlugin()
]
```

Webpack 구성 예제

- 프로젝트 디렉터리 생성
- package.json 생성
  - npm init



- webpack 설치

```
[전역 설치]
npm install -g webpack (Windows)
sudo  npm install -g webpack (macOS)

[ 개발 의존성 설치 ]
npm install --save-dev webpack
```

Webpack 구성 예제

바벨 트랜스파일러 설치

```
npm install --save-dev babel babel-core babel-loader babel-preset-es2015
```

.babelrc 파일 추가

```
{
   "presets" : [ "es2015" ]
}
```

webpack.config.js 파일 작성

```
var webpack = require('webpack');
module.exports = {
    entry: {
      main: __dirname + '/src/index.js'
    },
    output: {
        path: __dirname + '/public/dist/',
        filename: '[name].js',
        publicPath : '/dist'
    },    module: {
        rules: [
            {
                test:/₩.js$/,
                loader: 'babel-loader',
                exclude: /node_modules/
            }
        ]
    },
    plugins : [
        new webpack.optimize.UglifyJsPlugin()
    ]
}
```

Webpack 구성 예제

package.json 수정

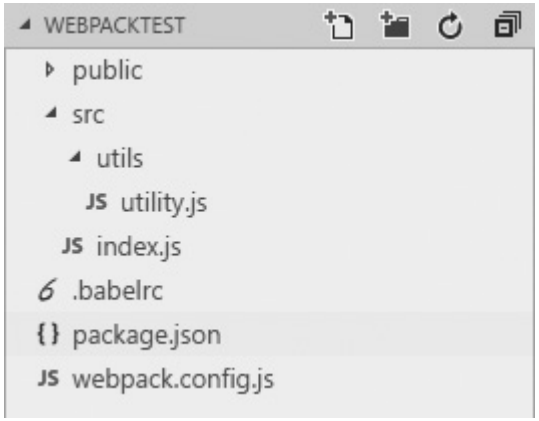
- npm run 명령어로 트랜스파일할 수 있도록…

```
{
  "name": "webpacktest",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "build": "webpack",
    "test": "echo ₩"Error: no test specified₩" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "babel": "^6.23.0",
    "babel-core": "^6.24.1",
    "babel-loader": "^7.0.0",
    "babel-preset-es2015": "^6.24.1",
    "webpack": "^2.5.1"
  }
}
```

Webpack 구성 예제

src/utils/utility..js 작성

```
let calc = {
    add(x,y) {
        return x+y;
    },
    multiply(x,y) {
        return x*y;
    }
}
export default calc;
```



src/index.js 작성

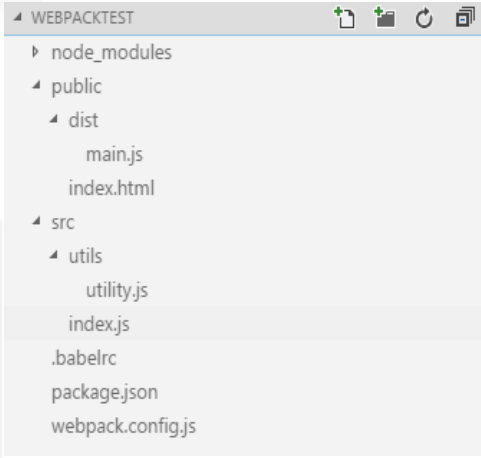
```
import calc from './utils/utility';
let x = 5;
let y = 5;
let str = `<h2>${x} + ${y} = ${calc.add(x,y)}</h2>`;
document.getElementById("app").innerHTML = str;
```

Webpack 구성 예제

테스트 페이지 작성

- public/index.html 작성

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>테스트 페이지</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
    </head>
    <body>
        <div id="app"></div>
    </body>
    <script src="dist/main.js"></script>
</html>
```



번들링후 실행!!

```
C:\dev\workspace\vuejs\webpacktest>npm run build

> webpacktest@1.0.0 build C:\dev\workspace\vuejs\webpacktest
> webpack

Hash: d864be063636b262e048
Version: webpack 2.5.1
Time: 642ms
  Asset       Size  Chunks             Chunk Names
main.js  841 bytes       0  [emitted]  main
   [0] ./src/utils/utility.js 249 bytes {0} [built]
   [1] ./src/index.js 343 bytes {0} [built]

C:\dev\workspace\vuejs\webpacktest>
```

Webpack 개발 서버 설정

- **앞절까지 설정한 것만으로도 번들링은 가능하지만...**
  - 개발중에 코드가 변경될때마다 매번 새롭게 webpack 명령을 수행해야...
- **HMR**
  - Hot Module Replacement
  - 코드가 변경되면 자동으로 번들링하고 브라우저를 갱신시킴
- **설치 및 package.json 변경**

```
npm install --save-dev webpack-dev-server
```

```
{
  ......
  "scripts": {
    "build": "webpack",
    "start": "node_modules/.bin/webpack-dev-server --open --hot",
    "test": "echo ₩"Error: no test specified₩" && exit 1"
  },
  ......
}
```

Webpack 개발 서버 설정

webpack.config.js 파일 수정

```
var webpack = require('webpack');
module.exports = {
    ……
    devServer: {
        contentBase: './public/',
        port: 3000,
        historyApiFallback:true
    }
}
```

- contentBase : 웹서버 루트로 지정할 디렉터리 경로 설정
- port : 포트 번호. 기본값은 8080
- historyApiFallback : URI가 존재하지 않을 때 index.html로 자동 이동시킬 지 여부 지정

실행

- npm run start
- 실행 후 코드를 변경하고 저장하면 즉시 반영됨.

Vue-CLI 보일러플레이트

- **simple 템플릿**
  - vue init simple [프로젝트명]
  - 단일 HTML 형태
- **Vue-CLI가 제공하는 Webpack 기반 템플릿**
  - webpack
  - webpack-simple
  - pwa
  - 각 템플릿의 특징은 1장에서 이미 다루었음!!

Vue-CLI 보일러플레이트

webpack-simple 기반 프로젝트 생성

- vue init webpack-simple webpacktest2
- package.json 파일

```
{
  ......
  "scripts": {
    "dev": "cross-env NODE_ENV=development webpack-dev-server --open --hot",
    "build": "cross-env NODE_ENV=production webpack --progress --hide-modules"
  },
  "dependencies": {
    "vue": "^2.2.1"
  },
  "devDependencies": {
    "babel-core": "^6.0.0",
    "babel-loader": "^6.0.0",
    "babel-preset-latest": "^6.0.0",
    "cross-env": "^3.0.0",
    "css-loader": "^0.25.0",
    "file-loader": "^0.9.0",
    "vue-loader": "^11.1.4",
    "vue-template-compiler": "^2.2.1",
    "webpack": "^2.2.0",
    "webpack-dev-server": "^2.2.0"
  }
}
```

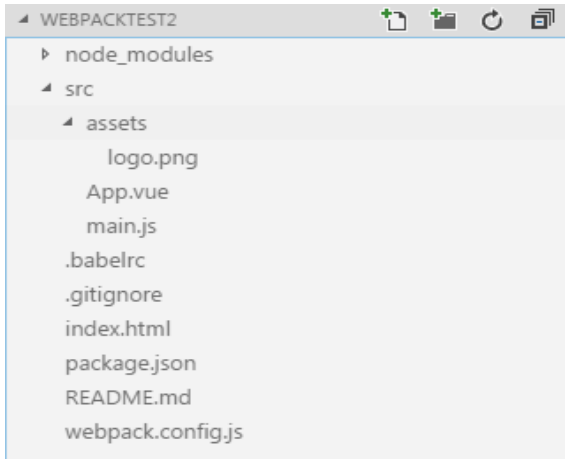
Vue-CLI 보일러플레이트

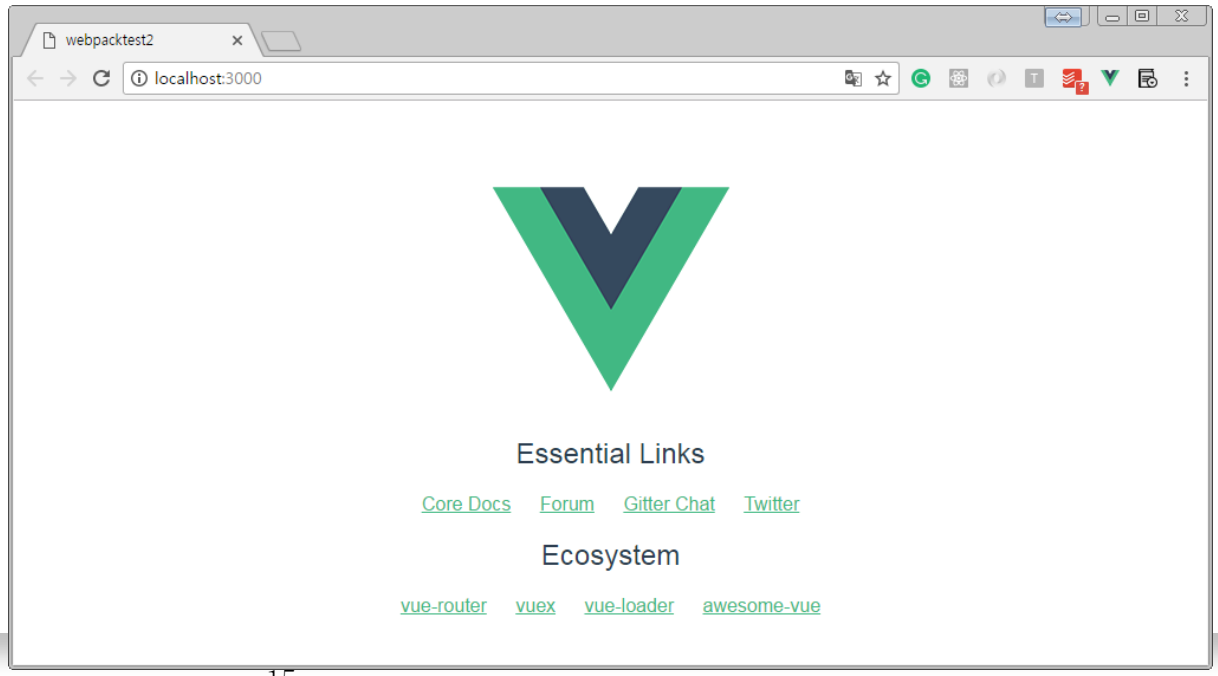
라이브러리 기능

- cross-env
  - 여러 플랫폼에 걸쳐 환경 변수를 설정하고 사용하는 스크립트를 실행
- css-loader
  - 컴포넌트에서 css 파일을 파일명을 이용해 직접 import 구문으로 로드하여 사용하는 방법을 제공
- file-loader
  - 정적 자원을 import 구문을 이용해 코드로 로드하는 기능. 로드된 파일은 webpack에 의해 번들링됨.
- vue-loader
  - 확장자가 .vue인 단일 파일 컴포넌트파일을 트랜스파일하고 로드하는 기능을 수행함
- vue-template-compiler
  - vue 컴포넌트 내부의 템플릿 문자열을 트랜스파일함.

Vue-CLI 보일러플레이트

- **webpack-simple의 디렉터리, 파일 구성**
  - webpack 개발 서버를 이용해 HMR 기능 지원
- **실행**
  - npm run dev





Vue-CLI 보일러플레이트

PWA

PWA(Progressive Web Application)는 웹 환경에서 최상의 사용자 경험을 제공하기 위한 방법이라고 할 수 있으며 다음과 같은 특성을 가집니다.

- 신뢰성(Reliable) : 불안정한 네트워크 상태에서도 즉시 로드하여 브라우저에 앱의 화면이 정상적으로 나타나도록 할 수 있습니다. 내부적으로 서비스 워커(ServiceWorker)를 이용하기 때문입니다.
- 빠른 속도(Fast) : 로드하는데 오랜 시간이 소요될 때 사용자가 사이트에서 빠져나가버리는 경우가 많습니다. 이런 상황이 발생하지 않도록 모든 리소스를 최적으로 신속하게 브라우저에 제공하여 즉각적인 웹 환경을 구현할 수 있어야 합니다.
- 매력적(Engaging) : 앱스토어 없이 사용자의 홈 화면에 설치되고 제공될 수 있어야 합니다. 이를 위해 웹 앱매니페스트(web app manifest) 파일을 이용해 전체 화면의 사용자 경험을 제공합니다. 웹 푸시 알림을 이용해 사용자의 적극적인 앱 사용을 유도할 수도 있어야 합니다. 웹앱매니페스트는 앱의 아이콘, 화면에서의 런처, 배경색 등을 시작할 수 있는 JSON 파일을 일컫는 것입니다. 이를 이용해서 사이트 방문시 설치 배너도 표시할 수 있습니다.
- 앱과의 유사성(app-like) : 모바일 앱, 데스크톱 앱과 동일한 실행 방식과 사용자 경험을 제공하도록 개발해야 합니다.
- HTTPS 통신 : HTTPS를 사용하여 통신이 안전하게 이루어질 수 있어야 합니다.

서비스 워커는 PWA의 핵심 기술 중의 하나입니다. 서비스 워커는 브라우저 기반에서 백그라운드로 스크립트를 실행하는 기능을 제공하며 웹 페이지와는 독립적으로 작동합니다. UI와 직접적으로 관련이 없는 기능을 구현할 때 유용합니다. 웹페이지에서 네트워크 요청이 발생하면 서비스 워커는 요청을 가로채고 캐시가 존재하는지 여부를 조사합니다. 캐시가 존재하는 경우 즉시 로딩하여 신속하게 초기 화면이 나타날 수 있도록 합니다.

하지만 이러한 PWA에서 요구하는 기능은 IE와 같은 낮은 버전의 웹브라우저에서는 작동하지 않습니다. 그렇기 때문에 데스크톱 환경보다는 모바일 환경에서 각광받는 UI이며, 앞으로 더 발전되고 확산될 것으로 생각됩니다.

Vue-CLI 보일러플레이트

- **Webpack에 대한 더 자세한 내용은?**
  - https://webpack.js.org/configuration/
- **다른 webpack 기반 템플릿**
  - webpack-simple 템플릿은 간단한 Vue 애플리케이션을 개발할 때 주로 사용
  - webpack
    - vue-router 설정과 간단한 예제가 추가되어 있음
  - pwa
    - webpack 템플릿 + PWA(Progressive Web Application) 개발을 위한 설정 추가
    - service worker 사용