프론트엔드 기초 03 [JS, NPM, Webpack]

프론트엔드 날개달기 : Vuejs, React 배우기 전에 꼭 알아야 하는 지식

짐코딩 Gym Coding
2022.10.05

JS 유용한 Array API

map() 배열안에 적용

```
const numbers = [1, 2, 3, 4, 5, 6];
const result = numbers.map(function(number){
    return number *2;
});

console.log(result);
class Student {
    constructor(name, korean, english, math){
        this.name = name;
        this.korean = korean;
        this.english = english;
        this.math = math;
    }
}

const student1 = new Student("a", 65, 22, 33);
const student2 = new Student("b", 68, 42, 43);
const student3 = new Student("c", 61, 62, 53);
const student4 = new Student("d", 63, 22, 63);

const students = [student1,  student2,  student3,  student4];

console.log(
    "영어점수",
    students.map((student) => student.english)
);
console.log(
    "이름",
    students.map((student) => student.name)
);
```

some()

배열중 하나만 조건이 맞으면 true 반환

every()

배열 요소가 모두 참일 때 true 반환

```
console.log(
    "영어 점수 20 이상이 존재?",
    students.some((student)=> student.english >=20)
);
console.log(
    "영어 점수  모두 30 이상인가?",
    students.every((student)=> student.english >=20)
);
```

filter()

필터링해 배열 리턴

```
console.log(
    "짝수 출력",
    numbers.filter(number=> number%2 === 0)
);
```

reduce()

배열의 각 요소에 리듀스 값을 넣어서 리턴

```
const fruits = ["사과", "딸기", "배", "딸기"];
```

중복 제거

```
const reduceFruits = fruits.reduce((acc, cur) => {
    if(acc.includes(cur) === false){
        acc.push(cur);
```

```
    }
    return acc;
}, []);

console.log(reduceFruits);
```

JS 모듈 시스템

모듈 → 여러개의 js를 합쳐놓은것이라고 생각하면 편함

모듈을 사용하지 않았을때 이름이 같은 변수가 변경되는 등 여러 문제가 생길 수 있으므로

모듈 시스템으로 스코프 단위로 관리를 한다.

type=”module” 로 적용하면 모듈의 스코프로 가진다.

장점

- 유지보수용이
- 네임스페이스화
- 재사용성

종류

AMD, CommonJS , UMD, ES Module

NPM node package manager

쉽게 자주쓰는 모듈을 받아주고 관리 해주는 도구

Node.js를 설치하면 NPM이 자동 설치 된다.

dayjs를 install 해서 사용해보자.

dayjs 모듈 다운

D:\frontend\nodeStudy\learn-npm>npm install dayjs

사용해보기 **& node** 런타임 실행

```
const dayjs = require('dayjs');
console.log(dayjs('2022-10-05').format('[YYYYescape] YYYY-MM-DDTHH:mm:ssZ[Z]'
));
console.log(dayjs('2022-10-05').format('YYYY-MM-DD' ));
```

```
D:\frontend\nodeStudy\learn-npm>node index.js
YYYYescape 2022-10-05T00:00:00+09:00Z
2022-10-05
```

package.json

프로젝트에 대한 정보를 갖고있는 파일

직접생성도 가능하고 npm -init 으로 생성도 가능하다

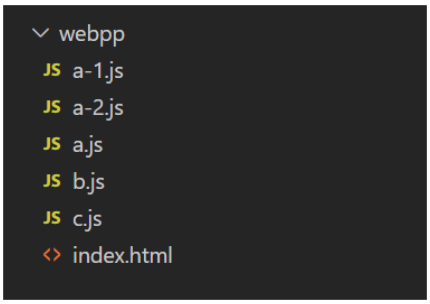
```
{
  "name": "learn-npm",          프로젝트 이름
  "version": "1.0.0",           버전
  "description": "npm 실습 프로젝트",   프로젝트 설명
  "main": "index.js",       프로젝트 기본 진입점
  "scripts": {  프로젝트에서 자주 사용되는 명령어 npm명령어로 바로 사용가능
    "test": "echo \"Error: no test specified\" && exit 1"

  },
  "author": "",
  "license": "ISC",
  "dependencies": {  프로젝트에서 사용하는 모듈을 기술하는 부분
    "dayjs": "^1.11.5"
  }
}
```

```
"devDependencies": {  프로젝트에서 개발시 사용하는 모듈을 기술하는 부분
  "nodemon": "^2.0.20" 버전[ MAJOR, MINOR, PATCH ]
}
```

package-lock.json 은 각 모듈의 의존관계가 기술되어있다.

웹팩 **webpack**

여러 모듈을 하나로 묶어주는 번들링을 수행해주는 번들러




```
    <script type="module">
       import a_number from './a.js';
       import b_number from './b.js';
       import c_number from './c.js';
       console.log('number ', a_number);
    </script>
```

전부 불러와 사용

이제 웹팩을 깔아서 사용한다면

```
D:\frontend\nodeStudy\webpp>npm install --save-dev webpack-cli
```

설치 후

npx명령어로 실행한다.

우선 index.js 를 생성해 하나로 묶어주고

index.js

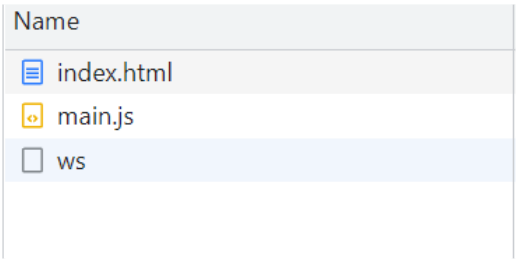
```
import a_number from './a.js';
import b_number from './b.js';
import c_number from './c.js';
```

```
D:\frontend\nodeStudy\webpp>npx webpack --entry .\src\index.js
--output-path .\dist
```

index.js 기준으로 번들링해 dist폴더에 저장된다.

```
    <script type="module" src="./dist/main.js"></script>
```

이제 번들링한 js를 넣으면 바로실행되고



깔끔하게 하나만 들어오게 된다.

→ 번들링 `npx webpack --entry .\src\index.js --output-path .\dist`

webpack.config.js 설정폴더를 만들어서 npx webpack으로 간단히 사용가능하고

npm 을 사용해 package.json에

```
  "scripts": {
    "build" : "webpack",
```

을 선언해 놓고 npm run build 로 간단히 번들링된 js를 생성한다.