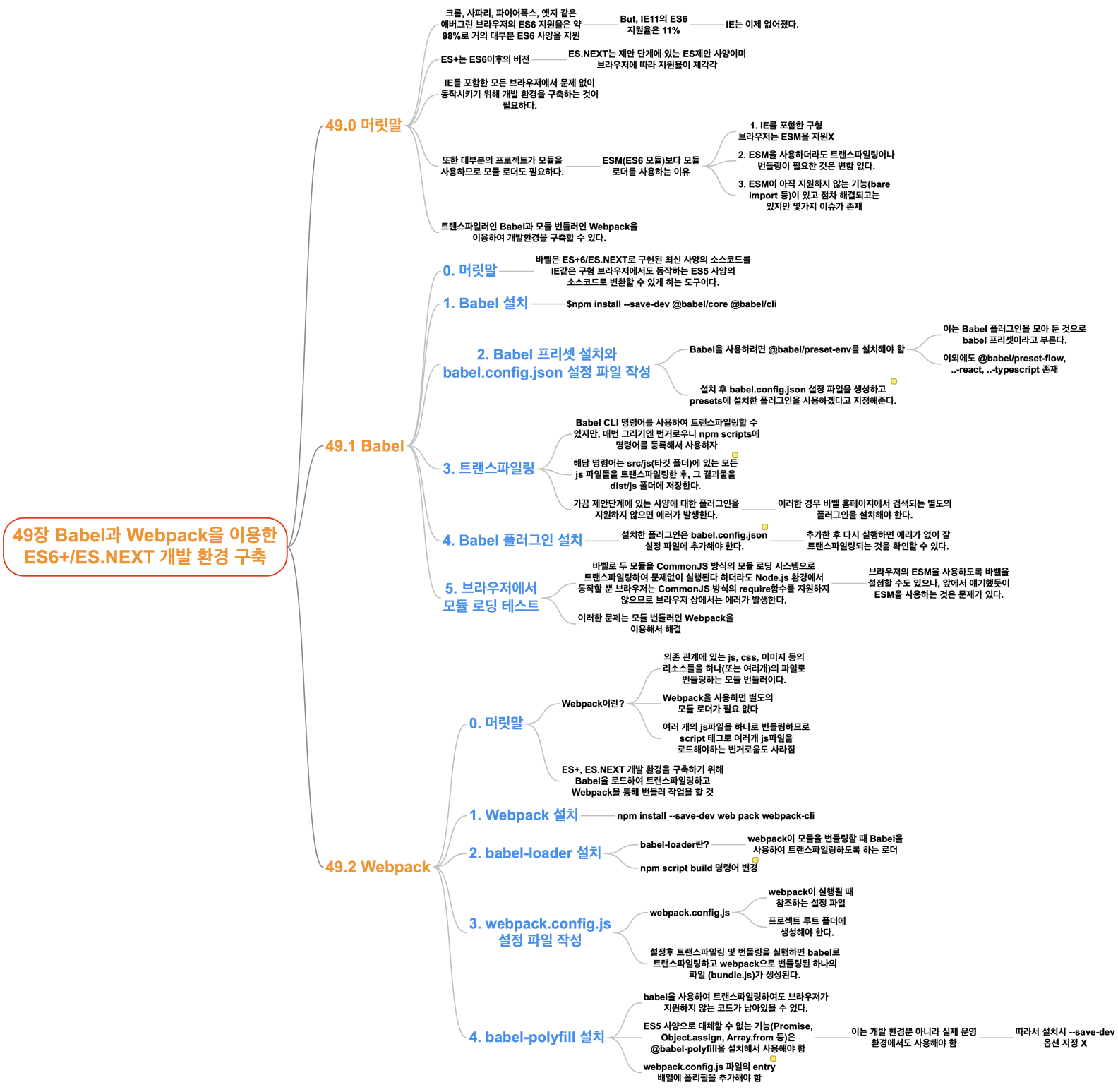
49장 Babel과 Webpack을 이용한 ES6+/ES.NEXT 개발 환경 구축
49.0 머릿말
크롬, 사파리, 파이어폭스, 엣지 같은 에버그린 브라우저의 ES6 지원율은 약 98%로 거의 대부분 ES6 사양을 지원
But, IE11의 ES6 지원율은 11%
IE는 이제 없어졌다.
ES+는 ES6이후의 버전
ES.NEXT는 제안 단계에 있는 ES제안 사양이며 브라우저에 따라 지원율이 제각각
IE를 포함한 모든 브라우저에서 문제 없이 동작시키기 위해 개발 환경을 구축하는 것이 필요하다.
또한 대부분의 프로젝트가 모듈을 사용하므로 모듈 로더도 필요하다.
ESM(ES6 모듈)보다 모듈 로더를 사용하는 이유
1. IE를 포함한 구형 브라우저는 ESM을 지원X
2. ESM을 사용하더라도 트랜스파일링이나 번들링이 필요한 것은 변함 없다.
3. ESM이 아직 지원하지 않는 기능(bare import 등)이 있고 점차 해결되고는 있지만 몇가지 이슈가 존재
트랜스파일러인 Babel과 모듈 번들러인 Webpack을 이용하여 개발환경을 구축할 수 있다.
49.1 Babel
0. 머릿말
바벨은 ES+6/ES.NEXT로 구현된 최신 사양의 소스코드를 IE같은 구형 브라우저에서도 동작하는 ES5 사양의 소스코드로 변환할 수 있게 하는 도구이다.
1. Babel 설치
$npm install --save-dev @babel/core @babel/cli
2. Babel 프리셋 설치와 babel.config.json 설정 파일 작성
Babel을 사용하려면 @babel/preset-env를 설치해야 함
이는 Babel 플러그인을 모아 둔 것으로 babel 프리셋이라고 부른다.
이외에도 @babel/preset-flow, ..-react, ..-typescript 존재
설치 후 babel.config.json 설정 파일을 생성하고 presets에 설치한 플러그인을 사용하겠다고 지정해준다.
3. 트랜스파일링
Babel CLI 명령어를 사용하여 트랜스파일링할 수 있지만, 매번 그러기엔 번거로우니 npm scripts에 명령어를 등록해서 사용하자
해당 명령어는 src/js(타깃 폴더)에 있는 모든 js 파일들을 트랜스파일링한 후, 그 결과물을 dist/js 폴더에 저장한다.
가끔 제안단계에 있는 사양에 대한 플러그인을 지원하지 않으면 에러가 발생한다.
이러한 경우 바벨 홈페이지에서 검색되는 별도의 플러그인을 설치해야 한다.
4. Babel 플러그인 설치
설치한 플러그인은 babel.config.json 설정 파일에 추가해야 한다.
추가한 후 다시 실행하면 에러가 없이 잘 트랜스파일링되는 것을 확인할 수 있다.
5. 브라우저에서 모듈 로딩 테스트
바벨로 두 모듈을 CommonJS 방식의 모듈 로딩 시스템으로 트랜스파일링하여 문제없이 실행된다 하더라도 Node.js 환경에서 동작할 뿐 브라우저는 CommonJS 방식의 require함수를 지원하지 않으므로 브라우저 상에서는 에러가 발생한다.
브라우저의 ESM을 사용하도록 바벨을 설정할 수도 있으나, 앞에서 얘기했듯이 ESM을 사용하는 것은 문제가 있다.
이러한 문제는 모듈 번들러인 Webpack을 이용해서 해결
49.2 Webpack
0. 머릿말
Webpack이란?
의존 관계에 있는 js, css, 이미지 등의 리소스들을 하나(또는 여러개)의 파일로 번들링하는 모듈 번들러이다.
Webpack을 사용하면 별도의 모듈 로더가 필요 없다
여러 개의 js파일을 하나로 번들링하므로 script 태그로 여러개 js파일을 로드해야하는 번거로움도 사라짐
ES+, ES.NEXT 개발 환경을 구축하기 위해 Babel을 로드하여 트랜스파일링하고 Webpack을 통해 번들러 작업을 할 것
1. Webpack 설치
npm install --save-dev web pack webpack-cli
2. babel-loader 설치
babel-loader란?
webpack이 모듈을 번들링할 때 Babel을 사용하여 트랜스파일링하도록 하는 로더
npm script build 명령어 변경
3. webpack.config.js 설정 파일 작성
webpack.config.js
webpack이 실행될 때 참조하는 설정 파일
프로젝트 루트 폴더에 생성해야 한다.
설정후 트랜스파일링 및 번들링을 실행하면 babel로 트랜스파일링하고 webpack으로 번들링된 하나의 파일 (bundle.js)가 생성된다.
4. babel-polyfill 설치
babel을 사용하여 트랜스파일링하여도 브라우저가 지원하지 않는 코드가 남아있을 수 있다.
ES5 사양으로 대체할 수 없는 기능(Promise, Object.assign, Array.from 등)은 @babel-polyfill을 설치해서 사용해야 함
이는 개발 환경뿐 아니라 실제 운영 환경에서도 사용해야 함
따라서 설치시 --save-dev 옵션 지정 X
webpack.config.js 파일의 entry 배열에 폴리필을 추가해야 함