Babel로 es6 모듈을 commonJS와 ES6 module 모두 호환되게 하기

아래처럼 es6로 클래스를 만들고 babel compile 후 다른 곳에서 불러오는데 에러가 났다.

```
export default class A {}

var A  = require('AAA');
var a = new A();
// Uncaught TypeError: _AAA.A is not a constructor
```

`A`를 출력해봤더니 constructor function이 object에 담긴 상태로 모듈이 import되었다.

```
console.log(A);
// { default: [ Function: A ] }
```

원인은 Babel에 있었는데, Babel@6부터 commonJS의 `module.exports`를 더 이상 기본으로 export하지 않고 'default'라는 키로 export한다. 때문에 Babel@6으로 컴파일된 모듈을 불러오려면 아래의 방법 중 하나를 써야한다.

```
// AAA.js
export default class A {}
// -------
var A = require('AAA').default;
var a = new A();
// or
import A from 'AAA';
const a = new A();
```

또는

```
// A.js
export class A {}
// -------
var A = require('AAA');
var a = new A();
// or
import { A } from 'AAA';
const a = new A();
```

뭔가 깔끔하지 않아보인다면 'babel-plugin-add-module-exports'을 설치하고 `.babelrc`에 플러그인으로 추가하자.

```
npm install babel-plugin-add-module-exports --save-dev
```

```
// .babelrc
{
  "presets": ["env"],
  "plugins": [
    "add-module-exports"
  ]
}
```

그러면 아래와 같이 쓸 수 있다.

```
// A.js
export default class A {}
// -------
var A = require('AAA');
var a = new A();
// or
import A from 'AAA';
const a = new A();
```

May 21, 2017