ES6 함수 추가기능 (8월 6일)

배정	
상태	8월
속성	@2021년 8월 5일 오후 11:19
언어	

화살표함수, 구조분해할당, rest parameter , let , const 추가, class, spread 연산자(...), 템플릿 리터럴,

ES6 함수의 구분

이름	constructor	prototype	super	arguments
일반 함수(Normal)	O	O	X	O
메서드(Method)	X	X	O	O
화살표 함수(Arrow)	X	X	X	X

ES6의 메서드 특징

- 메서드 축약 표현으로 정의된 함수만을 의미함.

```
const obj = {
  x: 1,
  // foo는 메서드이다.
  foo() { return this.x; },
  // bar에 바인딩된 함수는 메서드가 아닌 일반 함수이다.
  bar: function() { return this.x; }
};

console.log(obj.foo()); // 1
console.log(obj.bar()); // 1
```

- 인스턴스를 생성 할 수 없는 non-constructor다.

  축약된 함수는 인스턴스를 생성할수 없고, 일반함수는 인스턴스 생성 가능하다.

```
new obj.foo(); // -> TypeError: obj.foo is not a constructor
new obj.bar(); // -> bar {} 일반함수이기때문에 생성이됨.
```

```
// obj.foo는 constructor가 아닌 ES6 메서드이므로 prototype 프로퍼티가 없다.
obj.foo.hasOwnProperty('prototype'); // -> false

// obj.bar는 constructor인 일반 함수이므로 prototype 프로퍼티가 있다.
obj.bar.hasOwnProperty('prototype'); // -> true
```

- 바인딩한 객체를 가리키는 내부슬롯을 갖는다.(super)

super를 호출하면 수퍼클래스의 constructor(상위 클래스의 constructor) 를 호출함.

화살표 함수 (⇒ , arrow function)

화살표함수는 콜백 함수내부에서 this가 전역객체를 가리키는 문제를 해결하기위한 대안으로 유용함.

- 중괄호를 생략할 수가없음.
- 표현이 간결하고 가독성이좋음 .
- 일반함수의 기능을 간략화하여 this도 편리하게 설계되었음.
- 인스턴스를 생성할 수 없음. 프로토타입의 프로토타입도 생성하지않음.

화살표 함수 vs 일반 함수

Aa 비교	화살표 함수	일반 함수
인스턴스 생성여부	constructor	non-constructor
중복된 매개변수 이름 선언 가능 여부	불가능.	가능.(strict mode 에서는 중복된 매개변수 선언시 에러)
바인딩 여부	X	O