2022/5/11

1. Closure

   - 폐쇄된 공간 안의 데이터에 접근하기 위한 테크닉

   - 내부 함수가 외부 함수에 접근하기 위한 테크닉

   - 변수 은닉 and 메모리/코드 효율 극대화를 위해 사용 -> 클래스의 역할과 비슷하다고 볼 수 있음

   - 함수에 별명을 할당하는 alias의 역할과도 비슷하다고 생각함

   - 함수의 인자를 지정해서 변수에 할당할 수 있기 때문에 함수 내부의 내용을 드러내지 않을 수 있으며, 추후에 재사용 가능 및 코드의 반복을 줄일 수 있음

2. Map

- key-value 쌍을 가지는 객체 자료형
- set(key, value) 메소드를 이용하여 값 추가
- get(key) 메소드를 이용하여 value 접근
- has(key) 메소드를 이용하여 값이 있는지 확인
- delete(key) 메소드를 이용하여 값 제거
- size 메소드로 크기 확인 (length로는 확인 불가)
- keys() 메소드를 이용하여 key만을 가진 배열 반환
- values() 메소드를 이용하여 value만을 가진 배열 반환
- entries() 메소드를 이용하여 key-value 쌍을 가진 배열 반환
- key를 문자열 외에도 다른 타입으로 지정할 수 있음
- Object보다 성능이 좋음

3. Set

- 중복을 허용하지 않는 객체 자료형
- add(value) 메소드로 값 추가
- for ~ of … 문을 이용하여 원소 순회 가능
- delete(value) 메소드로 특정 값 제거
- clear() 메소드로 모든 값 제거
- filter, concat, has 등을 이용하여 두 Set의 교집합, 합집합, 차집합을 구할 수 있음

4. this.call() vs apply() vs bind()

4.1. this.call()

- function에 인자를 전달할 때 쉼표로 분리해서 전달 (ex. function.call(this자리에 들어갈 변수, a, b, c);

- 함수에 인자를 적용한 후 즉시 실행

4.2. this.apply()

- function에 인자를 전달할 때 배열로 전달 (ex. function.call(this자리에 들어갈 변수, [a, b, c]);

- 함수에 인자를 적용한 후 즉시 실행

4.3. this.bind()

- function에 인자를 전달하고 그게 적용된 후의 함수를 새롭게 반환

- 함수를 즉시 실행하지 않으므로 반환된 함수를 다른 변수에 저장 후 필요할 때 호출할 수 있음 (ex. let binded = function.bind(this자리에 들어갈 변수, a); -> binded(b, c);로 사용 가능)

- 추후에 이미 바인딩된 인자 대신 새로운 인자를 전달할 수도 있음 (ex. binded(this자리에 들어갈 변수2, a2);)

5. JSON

- 자바스크립트에서 객체를 표현하는 방식처럼 데이터를 표현한 파일 형식

- JSON.parse(json) 메소드를 이용하여 JSON을 Object로 변환

- JSON.stringify(obj) 메소드를 이용하여 Object를 JSON으로 변환

- 변수 은닉 and 메모리/코드 효율 극대화를 위해 사용 -> 클래스의 역할과 비슷하다고 볼 수 있음

- 함수에 별명을 할당하는 alias의 역할과도 비슷하다고 생각함

- 함수의 인자를 지정해서 변수에 할당할 수 있기 때문에 함수 내부의 내용을 드러내지 않을 수 있으며, 추후에 재사용 가능 및 코드의 반복을 줄일 수 있음