JavaScript

Reflect – 김근형 강사

reflect

- Reflect Object
  - Reflect는 자바스크립트 작업을 중간에서 가로챌 수 있는 built-in 객체다.
  - method들은 proxy handler들과 같다.
  - Reflect 오브젝트에 constructor가 없으므로 new 연산자로 인스턴스를 생성할 수 없다.
  - 그러므로 Reflect는 new 없이 인스턴스를 생성하지 않고 Reflect.get()과 같이 사용한다.
  - Reflect 오브젝트의 모든 메서드는 정적(static) 메서드이다.

reflect

- get()
  - target 오브젝트에서 프로퍼티 값을 반환한다

구분	타입	데이터(값)
형태		Reflect.get()
파라미터	object	Target, 대상 오브젝트
	String	Key, 프로퍼티 키
	object	(선택). receiver, this로 참조할 오브젝트
반환	any	프로퍼티 값

reflect

- get()
  - 맨 마지막 라인에서 this가 참조하는건 {event:"농구"} 이다.

```
let sports = {
  event: "축구",
  ground: "상암구장",
  get getGame(){
    return this.event + ":" + this.ground;
  }
};

console.log(Reflect.get(sports, "event"));
console.log(Reflect.get(sports, "getGame"));
console.log(Reflect.get(sports, "getGame", {event: "농구"}));
```

```
축구
축구:상암구장
농구:undefined
```

reflect

- get()

```
let sportsArray = ["축구", "농구"];

console.log(Reflect.get(sportsArray, 1));
console.log(Reflect.get(sportsArray, 5));
```

```
농구
undefined
```

reflect

- get()

```
let sports = {soccer: "축구"};
let newProxy = new Proxy(sports, {
  get(target, key, receiver){
    return target[key] + ",11명";
  }
});

console.log(Reflect.get(newProxy, "soccer"));
```

축구,11명

```
let sports = {soccer: "축구"};
let newProxy = new Proxy(sports, {
  get(target, key, receiver){
    return Reflect.get(target, key) + ":11명";
  }
});

console.log(Reflect.get(newProxy, "soccer"));
```

축구:11명

reflect

- set()
  - target 오브젝트에 프로퍼티 키와 값을 생성한다.

구분	타입	데이터(값)
형태		Reflect.set()
파라미터	object	Target, 대상 오브젝트
	String	Key, 프로퍼티 키
	any	Value, 프로퍼티 값
	Object	(선택). receiver, setter에서 this로 참조할 오브젝트
반환	Boolean	처리성공 true, 실패 false

reflect

- set()

```
let sportsObj = {};
console.log(Reflect.set(sportsObj, "soccer", "축구"));

console.log(sportsObj);
```

```
true
▼Object
    soccer: "축구"
  ▶__proto__: Object
```

```
let sportsObj = {
  set setGame(event){
    Reflect.set(sportsObj, event[0], this.player || event[1]);
  }
};

Reflect.set(sportsObj, "setGame", ["soccer"], {player: 11});
console.log(sportsObj);

Reflect.set(sportsObj, "setGame", ["baseball", 9]);
console.log(sportsObj);
```

```
▼Object
    baseball: 9
    soccer: 11
  ▼set setGame: ƒ setGame(event)
      arguments: (...)
      caller: (...)
      length: 1
      name: "set setGame"
    ▶__proto__: ƒ ()
      [[FunctionLocation]]: set-2.js:5
    ▶[[Scopes]]: Scopes[2]
  ▶__proto__: Object
```

reflect

- set()

```
let sportsArray = ["농구"];

Reflect.set(sportsArray, 1, "축구");
console.log(sportsArray);

Reflect.set(sportsArray, 0, "아이스하키");
console.log(sportsArray);
```

```
▼Array(2) i
    0: "아이스하키"
    1: "축구"
    length: 2
  ▶ __proto__: Array(0)
```

```
let sportsObj = {};
let newProxy = new Proxy(sportsObj, {
  set(target, key, value, receiver){
    Reflect.set(target, key, value);
  }
});

Reflect.set(newProxy, "baseball", "야구");
console.log(sportsObj);
```

```
▼Object i
    baseball: "야구"
  ▶ __proto__: Object
```

reflect

- has()
  - target 오브젝트에서 프로퍼티 키의 존재 여부를 반환한다.

구분	타입	데이터(값)
형태		Reflect.has()
파라미터	object	Target, 대상 오브젝트
	String	Key, 프로퍼티 키
반환	Boolean	존재하면 true, 아니면 false

reflect

- has()

```
let nameObj = {name: "이름"};
console.log(Reflect.has(nameObj, "name"));

function sports(){};
console.log(Reflect.has(sports, "hasOwnProperty"));
```

```
true
true
```

```
let sportsObj = {baseball: "야구"};
let newProxy = new Proxy(sportsObj, {
  has(target, key){
    Reflect.has(target, key);
  }
});

console.log(Reflect.has(newProxy, "baseball"));
```

```
false
```

reflect

- apply()
  - 함수를 호출한다

구분	타입	데이터(값)
형태		Reflect.apply()
파라미터	Function	Target, 호출할 함수 이름
	object	(선택) this, this로 참조할 오브젝트
	Array	(선택) argument, 호출된 함수로 넘겨 줄 파라미터 값
반환	any	호출된 함수에서 반환한 값

reflect

- apply()

```
function getValue(...values){
  return this.base + values.reduce(function(pre, cur){
    return pre + cur;
  });
}

console.log(Reflect.apply(getValue, {base: 100}, [10, 20, 30]));
```

```
160
```

```
let result = Reflect.apply(String.prototype.indexOf, "ABC", ["B"]);
console.log(result);
```

```
1
```

reflect

- apply()

```
function getValue(...values){
  return this.base + values.reduce(function(pre, cur){
      return pre + cur;
  });
};

let newProxy = new Proxy(getValue, {
  apply(target, thisObj, params) {
    return Reflect.apply(target, thisObj, params);
  }
});

console.log(Reflect.apply(newProxy, {base: 100}, [10, 20, 30]));
```

```
160
```

reflect

- construct()
  - 인스턴스를 생성하여 반환한다

구분	타입	데이터(값)
형태		Reflect.construct()
파라미터	Function	Target, 생성자 함수
	Array	(선택) args, 호출된 함수로 넘겨 줄 파라미터
	object	(선택) newTarget, 생성자 함수
반환	instance	생성한 인스턴스

reflect

- construct()

```
class Sports{
  constructor(ground){
    this.ground = ground;
  }
};
let obj = Reflect.construct(Sports, ["상암구장"]);
console.log(obj.ground);
```

상암구장

reflect

- construct()

```
class Sports{
  constructor(ground){
    this.ground = ground;
  }
  getGround(){
    return this.ground;
  }
};

class Soccer{
  getGround(){
    return "Soccer.getGround() 사용";
  }
};

let obj = Reflect.construct(Sports, ["상암구장"], Soccer);
console.log(obj.getGround());
```

```
Soccer.getGround() 사용
```

reflect

- construct()

```
class Sports{
  constructor(...values){
    this.values = values;
  }
  getValues(){
    return this.values;
  }
};

let newProxy = new Proxy(Sports, {
  construct(target, params, proxy){
    return Reflect.construct(target, params);
  }
});

let obj = Reflect.construct(newProxy, ["축구", "상암구장"]);
console.log(obj.getValues());
```

```
▼Array(2) i
    0: "축구"
    1: "상암구장"
    length: 2
  ▶ __proto__: Array(0)
```

reflect

- defineProperty()
  - target 오브젝트에 프로퍼티를 추가하거나 프로퍼티 값을 변경한다

구분	타입	데이터(값)
형태	Reflect	Reflect.defineProperty()
파라미터	Object	Target, 대상 오브젝트
	String	Key 추가/변경할 프로퍼티 키
	Object	추가/변경할 attribute(디스크립터)
반환	Boolean	처리 성공 true, 실패 false

reflect

- defineProperty()

```
let sportsObj = {
  event: "축구"
};

let result = Reflect.defineProperty(sportsObj, "sports", {
  get(){
    return this.event;
  }
});
console.log(result);
console.log(sportsObj.sports);
```

```
true
축구
```

reflect

- deleteProperty()
  - target 오브젝트에서 프로퍼티를 삭제한다

구분	타입	데이터(값)
형태		Reflect.deleteProperty()
파라미터	object	Target, 대상 오브젝트
	String	Key, 삭제할 프로퍼티 키
반환	Boolean	처리 성공 true, 아니면 false

reflect

- deleteProperty()

```
let obj = {member: 11, ground: "상암"};
console.log(Reflect.deleteProperty(obj, "ground"));

console.log(Reflect.deleteProperty(obj, "ground"));

Object.freeze(obj);
console.log(Reflect.deleteProperty(obj, "member"));
```

```
true
true
false
```

reflect

- getOwnPropertyDescriptor()
  - target 오브젝트에서 프로퍼티의 디스크립터를 반환한다.

구분	타입	데이터(값)
형태		Reflect.getOwnPropertyDescriptor()
파라미터	object	Target, 대상 오브젝트
	String	Key, 프로퍼티 키
반환	object	프로퍼티 디스크립터

```
let sportsObj = {sports: "스포츠"};

let result = Reflect.getOwnPropertyDescriptor(sportsObj, "sports");
console.log(result);
```

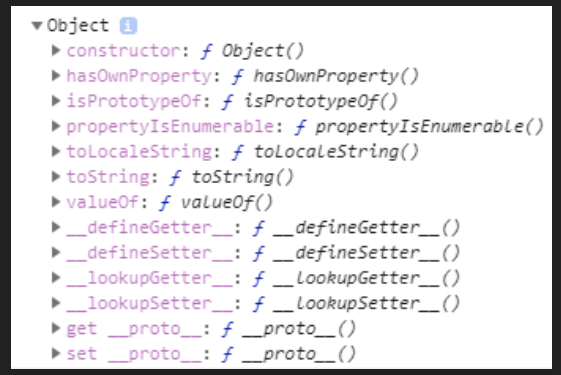
```
▼Object
    configurable: true
    enumerable: true
    value: "스포츠"
    writable: true
  ▶__proto__: Object
```

reflect

- getPrototypeOf()
  - target 오브젝트의 prototype(__proto__)에 연결된 프로퍼티를 반환한다.

구분	타입	데이터(값)
형태		Reflect.getPrototypeOf()
파라미터	object	Target, 대상 오브젝트
반환	object	오브젝트의 prototype(__proto__)에 연결된 프로퍼티

```
let obj = {sports: "스포츠"};
console.log(Reflect.getPrototypeOf(obj));
```



reflect

- setPrototypeOf()
  - target 오브젝트의 __proto__에 prototype을 설정한다

구분	타입	데이터(값)
형태		Reflect.setPrototypeOf()
파라미터	object	Target, 대상 오브젝트
	Object	Prototype, 설정할 오브젝트의 prototype
반환	Boolean	처리 성공 true, 실패 false

```
let Sports = function(){};
Sports.prototype.getGround = function(){
  return this.ground;
};

let groundObj = {ground: "상암구장"};
Reflect.setPrototypeOf(groundObj, Sports.prototype);

console.log(groundObj.getGround());
```

상암구장

reflect

- preventExtensions()
  - target 오브젝트에 프로퍼티 추가 금지를 설정한다

구분	타입	데이터(값)
형태		Reflect.preventExtension()
파라미터	object	Target, 대상 오브젝트
반환	Boolean	추가 금지 설정 성공 true, 아니면 false

```
let obj = {};
console.log(Reflect.preventExtensions(obj));

console.log(Reflect.defineProperty(obj, "baseball", {value: "야구"}));
```

```
true
false
```

reflect

- isExtensible()
  - target 오브젝트에 프로퍼티 추가 기능 여부를 반환한다.

구분	타입	데이터(값)
형태		Reflect.isExtension()
파라미터	object	Target, 대상 오브젝트
반환	Boolean	확장 가능 true, 아니면 false

```
let emptyObj = {};
Reflect.preventExtensions(emptyObj);

console.log(Reflect.isExtensible(emptyObj));
```

```
false
```

reflect

- ownKeys()
  - target 오브젝트에 프로퍼티키를 배열로 반환한다.

구분	타입	데이터(값)
형태		Reflect.ownKeys()
파라미터	object	Target, 대상 오브젝트
반환	Array	프로퍼티 키

```
let sportsObj = {
    soccer: "축구",
    baseball: "야구",
    [Symbol.for("one")]: 10
};
console.log(Reflect.ownKeys(sportsObj));
```