The Reflect API

Mark Zamoyta
SOFTWARE DEVELOPER

@markzamoyta

The Reflect API

- Object Construction
- Method Calls
- Prototypes
- Properties
- Property Extensions

Construction and Method Calls

```
console.log(typeof Reflect);
```

Question

What shows in the console?

Answer

object

```
Reflect.construct(target, argumentsList[, newTarget])
```

```
class Restaurant {
}

let r = Reflect.construct(Restaurant);
console.log(r instanceof Restaurant);
```

Question

What shows in the console?

Answer

true

```
class Restaurant {
    constructor(name, city) {
        console.log(`${name} in ${city}`);
    }
}

let r = Reflect.construct(Restaurant, ["Zoey's", "Goleta"]);
```

Q What shows in the console?

A Zoey's in Goleta

```
class Restaurant {
    constructor() {
        console.log(`new.target: ${new.target}`);
    }
}
function restaurantMaker() {
    console.log(`in restaurantMaker`);
}

let r = Reflect.construct(Restaurant,
                          ["Zoey's", "Goleta"], restaurantMaker);
```

Q What shows in the console?

A

```
new.target:
function restaurantMaker() {
 console.log(`in restaurantMaker`);
}
```

```
Reflect.apply(target, thisArgument, argumentsList)
```

```
class Restaurant {
    constructor() {
        this.id = 33;
    }
    show() {
        console.log(this.id);
    }
}
Reflect.apply(Restaurant.prototype.show, { id: 99 });
```

Q What shows in the console?

A 99

```
class Restaurant {
    constructor() {
        this.id = 33;
    }
    show(prefix) {
        console.log(prefix + this.id);
    }
}
Reflect.apply(Restaurant.prototype.show, { id: 22 }, ['REST:']);
```

Q What shows in the console?

A REST:22

Reflect and Prototypes

```
Reflect.getPrototypeOf(target)
```

```
class Location {
    constructor() {
        console.log('constructing Location');
    }
}
class Restaurant extends Location {
}
console.log(Reflect.getPrototypeOf(Restaurant));
```

Q What shows in the console?

A

```
constructor() {
 console.log('constructing Location');
}
```

```
Reflect.setPrototypeOf(target, prototype)
```

```
class Restaurant {
}
let setup = {
    getId() { return 88; }
}

let r = new Restaurant();
Reflect.setPrototypeOf(r, setup);
console.log(r.getId());
```

Question

What shows in the console?

Answer

88

Reflect and Properties

```
Reflect.get(target, propertyKey[, receiver])
```

```
class Restaurant {
    constructor() {
        this.id = 8000;
    }
}

let r = new Restaurant();
console.log(Reflect.get(r, 'id'));
```

Question

What shows in the console?

Answer

8000

```
class Restaurant {
    constructor() {
        this._id = 9000;
    }
    get id() {
        return this._id;
    }
}

let r = new Restaurant();
console.log(Reflect.get(r, 'id', { _id: 88 }));
```

Q What shows in the console?

A 88

```
Reflect.set(target, propertyKey, value[, receiver])
```

```
class Restaurant {
    constructor() {
        this.id = 9000;
    }
}

let r = new Restaurant();
Reflect.set(r, 'id', 88);
console.log(r.id);
```

Question

What shows in the console?

Answer

88

```
class Restaurant {
    constructor() {
        this._id = 9000;
    }
    set id(value) {
        this._id = value;
    }
}

let r = new Restaurant();
let alt = { id: 88 };
Reflect.set(r, '_id', 88, alt);
console.log(r._id)
console.log(alt._id);
```

Question

What shows in the console?

Answer

9000
88

```
Reflect.has(target, propertyKey)
```

```
class Location {
    constructor() {
        this.city = 'Goleta';
    }
}
class Restaurant extends Location {
    constructor() {
        super();
        this.id = 9000;
    }
}

let r = new Restaurant();
console.log(Reflect.has(r, 'id'));
console.log(Reflect.has(r, 'city'));
```

Question

What shows in the console?

Answer

true
true

```
Reflect.ownKeys(target)
```

```
class Location {
    constructor() {
        this.city = 'Goleta';
    }
}
class Restaurant extends Location {
    constructor() {
        super();
        this.id = 9000;
    }
}

let r = new Restaurant();
console.log(Reflect.ownKeys(r));
```

Question

What shows in the console?

Answer

["city", "id"]

```
Reflect.defineProperty(target, propertyKey, attributes)
```

```
class Restaurant {
}

let r = new Restaurant();

Reflect.defineProperty(r, 'id', {
    value: 2000,
    configurable: true,
    enumerable: true
});

console.log(r['id']);
```

Q What shows in the console?

A 2000

```
Reflect.deleteProperty(target, propertyKey)
```

```
let rest = {
    id: 2000
};

console.log(rest.id);

Reflect.deleteProperty(rest, 'id');

console.log(rest.id);
```

Question

What shows in the console?

Answer

2000
undefined

```
Reflect.getOwnPropertyDescriptor(target, propertyKey)
```

```
let rest = {
    id: 2000
};

let d = Reflect.getOwnPropertyDescriptor(rest, 'id');

console.log(d);
```

Q What shows in the console?

A {configurable: true, enumerable: true, value: 2000, writable: true}

Reflect and Property Extensions

```
Reflect.preventExtensions(target)
```

```
let rest = {
    id: 2000
};

rest.location = 'Goleta';

console.log(rest.location);
```

Question

What shows in the console?

Answer

Goleta

```
let rest = {
    id: 2000
};

Reflect.preventExtensions(rest);
rest.location = 'Goleta';

console.log(rest.location);
```

Question

What shows in the console?

Answer

undefined

```
Reflect.isExtensible(target)
```

```
let rest = {
    id: 2000
};

console.log(Reflect.isExtensible(rest));

Reflect.preventExtensions(rest);

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

Q What shows in the console?

A
true
false

Summary

Object Construction

Reflect.construct()

```
class Restaurant {
}

let r = Reflect.construct(Restaurant);
console.log(r instanceof Restaurant);
```

Summary

Method Calls

Reflect.apply()

```
class Restaurant {
    constructor() {
        this.id = 33;
    }
    show() {
        console.log(this.id);
    }
}
Reflect.apply(Restaurant.prototype.show, { id: 99 });
```

Summary

Prototypes

Reflect.getPrototypeOf()

Reflect.setPrototypeOf()

```
class Restaurant {
}
let setup = {
    getId() { return 88; }
}

let r = new Restaurant();
Reflect.setPrototypeOf(r, setup);
console.log(r.getId());
```

Summary

Properties

Reflect.get()

Reflect.set()

Reflect.has()

Reflect.ownKeys()

Reflect.defineProperty()

Reflect.deleteProperty()

Reflect.getOwnPropertyDescriptor()

Summary

Property Extensions

Reflect.preventExtensions()

Reflect.isExtensible()

```
let rest = {
    id: 2000
};

Reflect.preventExtensions(rest);
rest.location = 'Goleta';

console.log(rest.location);
```