The Proxy API

Defining Proxies

Available Traps

Get by Proxy

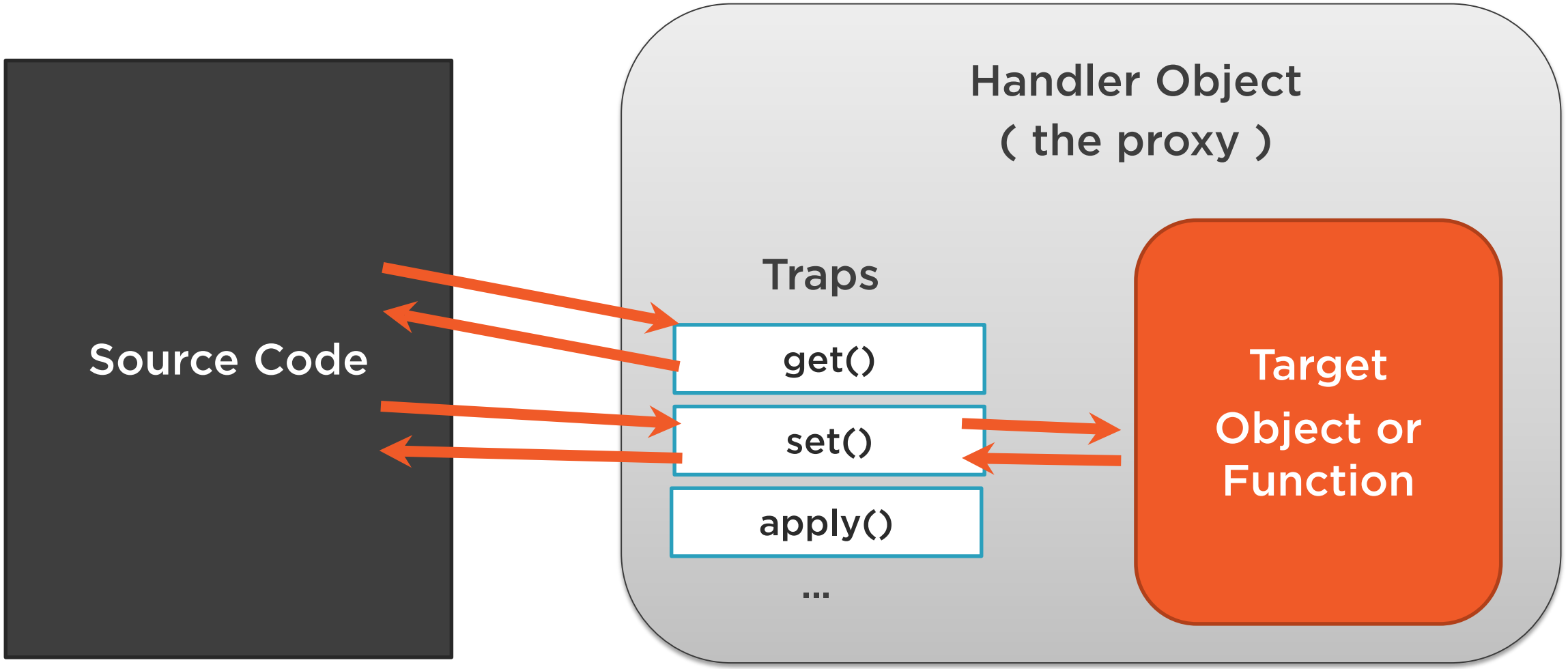
Calling Functions by Proxy

A Proxy as a Prototype

Revocable Proxies

Proxies Defined

Proxy Terminology



Available Traps

handler.construct()

handler.apply()

handler.getPrototypeOf()
handler.setPrototypeOf()

handler.get()
handler.set()
handler.has()
handler.ownKeys()

handler.defineProperty()
handler.deleteProperty()
handler.getOwnPropertyDescriptor()

handler.preventExtensions()
handler.isExtensible()

Untrappable Object Usage

Comparisons (== and ===)

typeof and instanceof

Operations (target + " ")

String(target)

Get by Proxy

```
function Employee () {
    this.name = 'Milton Waddams';
    this.salary = 0;
}
var e = new Employee();

var p = new Proxy(e, {
    get: function (target, prop, receiver) {
        return "Attempted access: " + prop;
    }
});

console.log(p.salary);
```

Question

What shows in the console?

Answer

Attempted access: salary

```
function Employee () {
    this.name = 'Milton Waddams';
    this.salary = 0;
}
var e = new Employee();

var p = new Proxy(e, {
    get: function (target, prop, receiver) {
        return Reflect.get(target, prop,
                           receiver);
    }
});

console.log(p.salary);
```

Question

What shows in the console?

Answer

0

```
function Employee () {
    this.name = 'Milton Waddams';
    this.salary = 0;
}
var e = new Employee();

var p = new Proxy(e, {
    get: function (target, prop, receiver) {
        if (prop === 'salary')
            return 'Denied';
        return Reflect.get(target, prop,
                           receiver);
    }
});

console.log(p.salary);
console.log(p.name);
```

Question

What shows in the console?

Answer

Denied
Milton Waddams

Calling Functions by Proxy

```
function getId() {
    return 55;
}

var p = new Proxy(getId, {
    apply: function (target, thisArg, argumentsList) {
        return Reflect.apply(target, thisArg, argumentsList);
    }
});

console.log( p() );
```

Q What shows in the console?

A 55

A Proxy as a Prototype

```
var t = {
    tableId: 99
}
var p = new Proxy({}, {
    get: function (target, prop, receiver) {
        return 'Property ' + prop + ' doesn\'t exist...';
    }
});

Object.setPrototypeOf(t, p);

console.log(t.tableId);
console.log(t.size);
```

Q What shows in the console?

A
99
Property size doesn't exist

Revocable Proxies

```
Proxy.revocable(...)
```

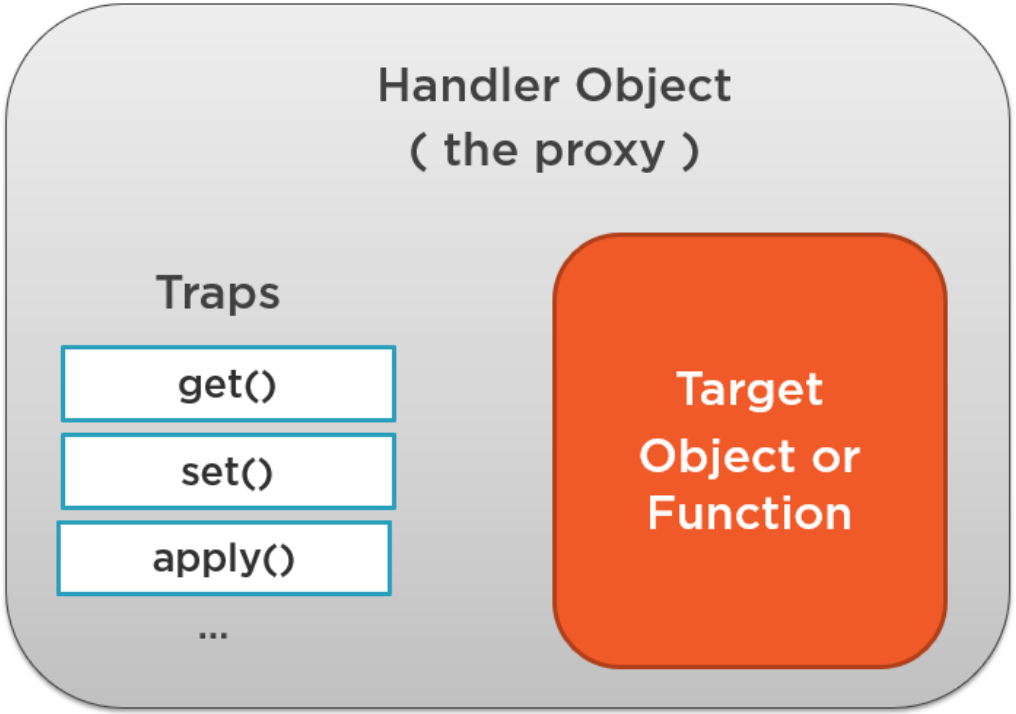
```
var t = {
    tableId: 99
}

let { proxy, revoke } = Proxy.revocable(t, {
    get: function (target, prop, receiver) {
        return Reflect.get(target, prop, receiver) + 100;
    }
});

console.log(proxy.tableId);
revoke();
console.log(proxy.tableId);
```

Q What shows in the console?

A
99
Property size doesn't exist

Summary

Defining Proxies



Summary

Traps

handler.construct()

handler.apply()

handler.getPrototypeOf()
handler.setPrototypeOf()

handler.get()
handler.set()
handler.has()
handler.ownKeys()

handler.defineProperty()
handler.deleteProperty()
handler.getOwnPropertyDescriptor()

handler.preventExtensions()
handler.isExtensible()

Summary

Get

```
function Employee () {
    this.name = 'Milton Waddams';
    this.salary = 0;
}
var e = new Employee();

var p = new Proxy(e, {
    get: function (target, prop, receiver) {
        return "Attempted access: " + prop;
    }
});

console.log(p.salary);
```

Summary

Calling Functions

```
function getId() {
    return 55;
}

var p = new Proxy(getId, {
    apply: function (target, thisArg, argumentsList) {
        return Reflect.apply(target, thisArg, argumentsList);
    }
});

console.log( p() );
```

Summary

A Proxy as a Prototype

```
var t = {
    tableId: 99
}
var p = new Proxy({}, {
    get: function (target, prop, receiver) {
        return 'Property ' + prop + ' doesn\'t exist...';
    }
});

Object.setPrototypeOf(t, p);

console.log(t.tableId);
console.log(t.size);
```

Summary

Revocable Proxies

```
var t = {
    tableId: 99
}

let { proxy, revoke } = Proxy.revocable(t, {
    get: function (target, prop, receiver) {
        return Reflect.get(target, prop, receiver) + 100;
    }
});

console.log(proxy.tableId);
revoke();
console.log(proxy.tableId);
```