Proxy Objects and Reflect in JavaScript

JavaScript's Proxy Object

Shawn Wildermuth

Instructor, Microsoft MVP, and Filmmaker

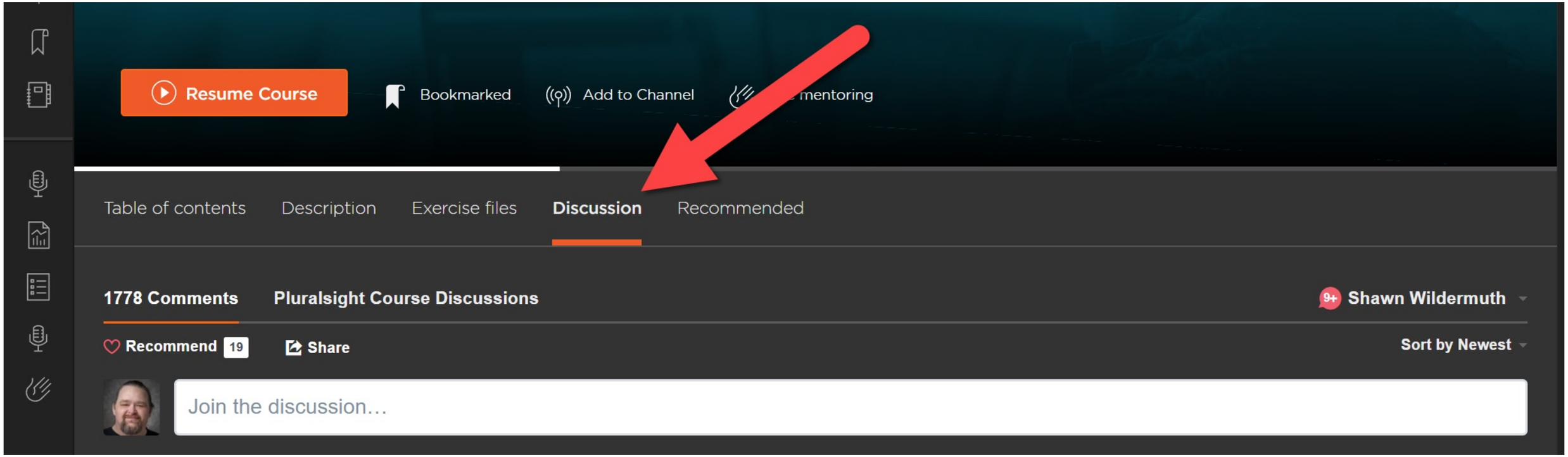
@shawnwildermuth | shawn.wildermuth.com

Course Overview

Proxy Objects and Reflect in JavaScript

- What Proxies are used for
- How Proxies work
- How to use them in common scenarios
- How to use Reflect to simplify Proxies

Questions During the Course?



Proxy Objects and Reflect in JavaScript

Version Check

Version Check

This course was created by using:

- ECMAScript 2015 and later
  - (e.g. JavaScript ES6 and later)

Version Check

This course is 100% applicable to:

- ECMAScript 2015 or later
  - (also known as JavaScript ES6)

Not Applicable

This course is NOT applicable to:

- ECMAScript 5 and earlier
  - (also known as JavaScript ES3 or ES5)

Module Overview

JavaScript's Proxy Object

- How JavaScript objects actually work
- Why you might need a Proxy
- Where Proxies work
- How to create a Proxy
- How to implement basic traps

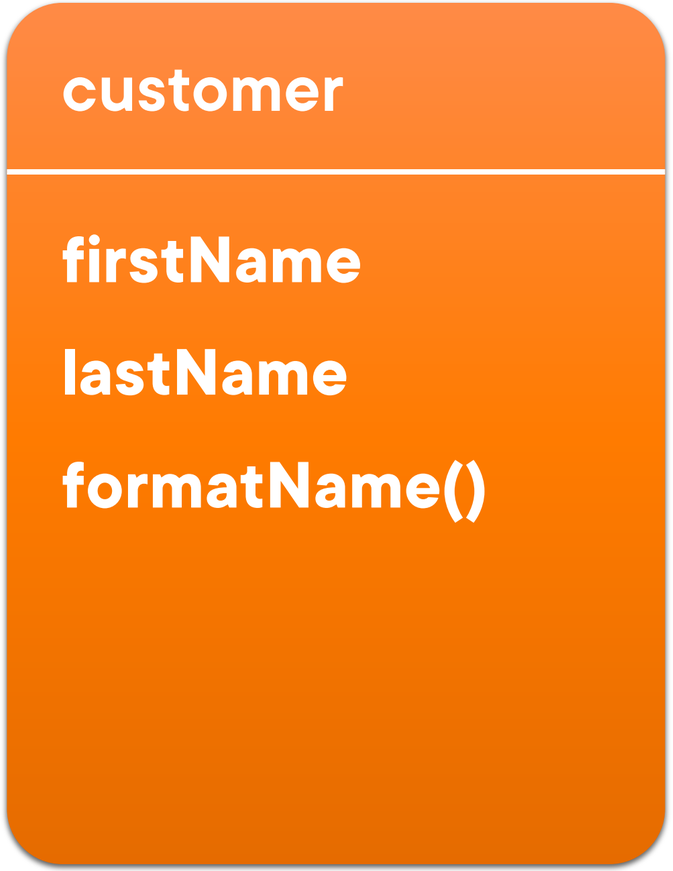
JavaScript Objects

What is a JavaScript Object Anyway?

```
const customer = {
  firstName: "",
  lastName: "",
  formatName() {...}
};
```

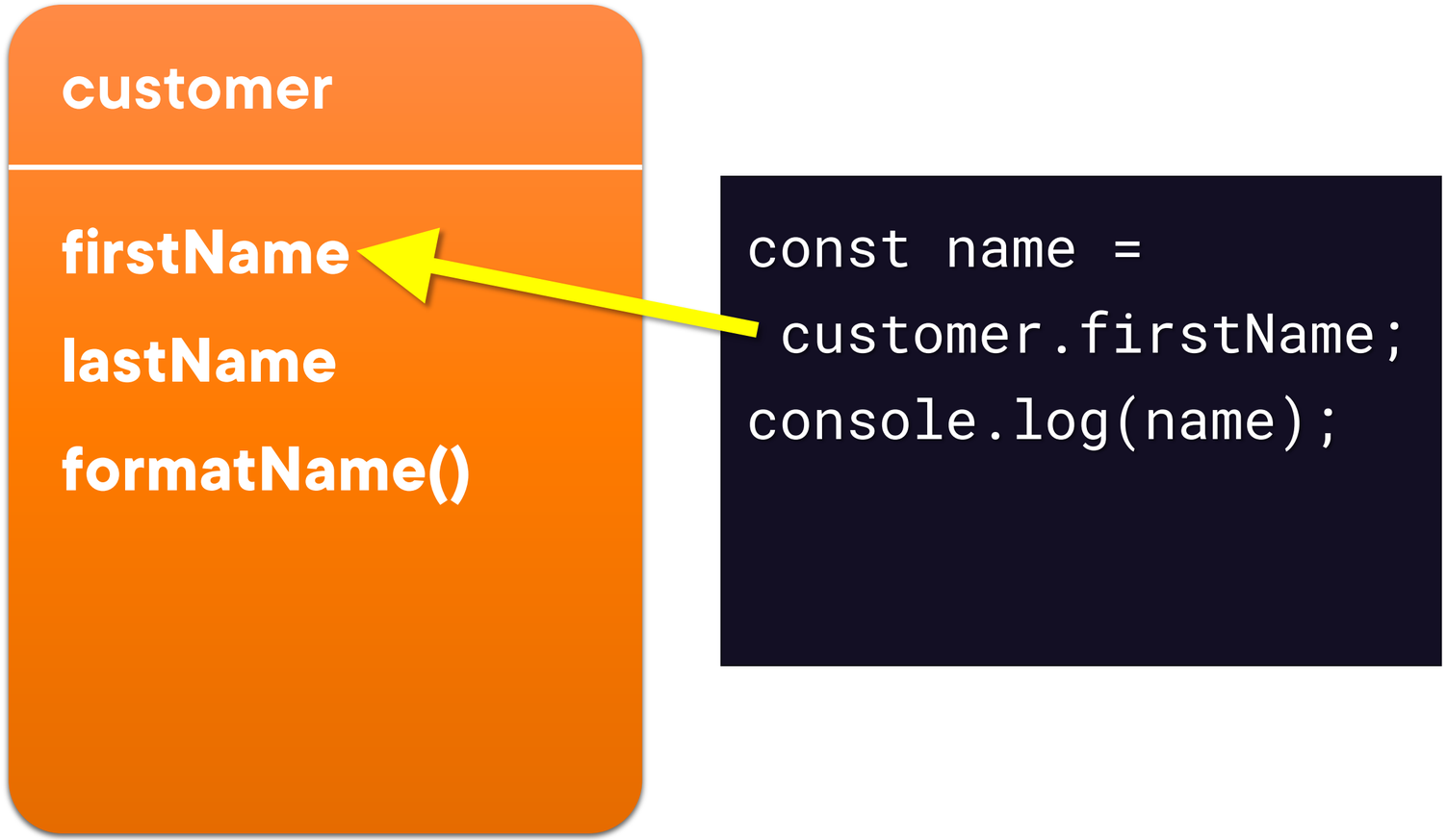
What is a JavaScript Object Anyway?

```
const customer = {
  firstName: "",
  lastName: "",
  formatName() {...}
};
```



What is a JavaScript Object Anyway?

```
const customer = {
  firstName: "",
  lastName: "",
  formatName() {...}
};
```




```
const name =
 customer.firstName;
console.log(name);
```

What is a JavaScript Object Anyway?

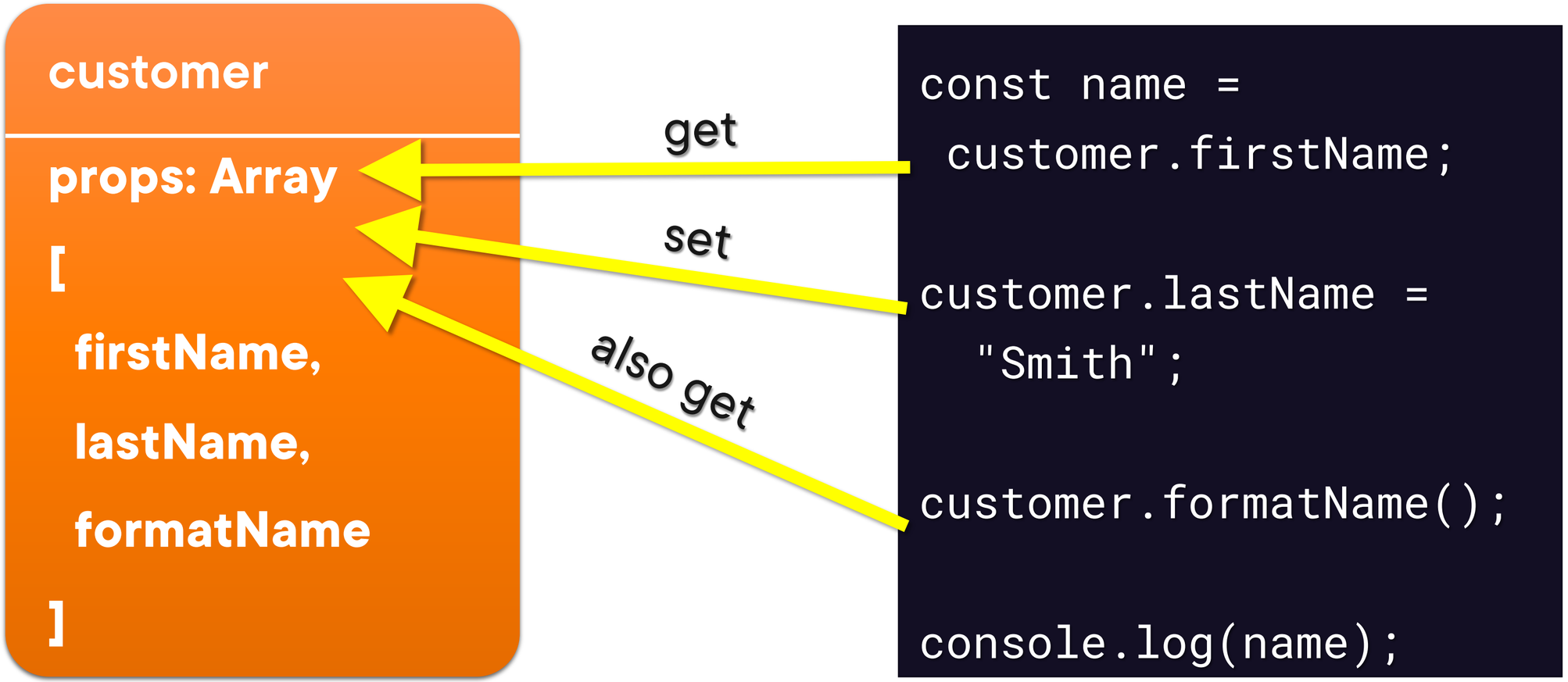
```
const customer = {
  firstName: "",
  lastName: "",
  formatName() {...}
};
```

What is a JavaScript Object Anyway?

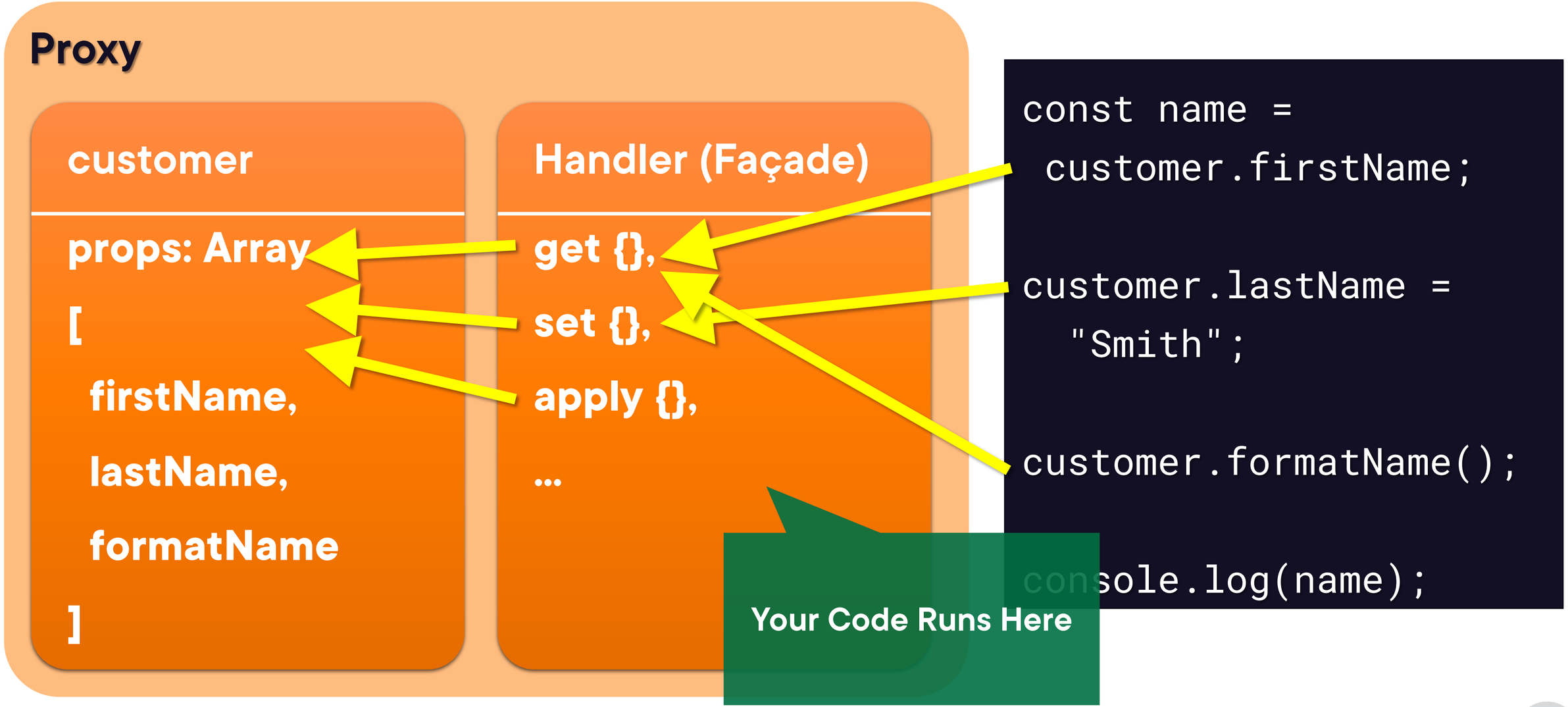
```
const customer = {
  firstName: "",
  lastName: "",
  formatName() {...}
};
```


```
const name =
 customer["firstName"];
console.log(name);
```

What is a JavaScript Object Anyway?



Then, What is a Proxy?



```
const product = {
  name: "",
  price: 0
};

const proxy = new Proxy(product);


proxy.name = "Nails";


out(proxy.name);   // "Nails"

out(product.name); // "Nails"
```

◀ A simple Object

◀ Create a Proxy

◀ Setting a value through the proxy

◀ Get a value as expected

◀ Underlying Object is changed too

Demo

Creating a Proxy

```
const number = 25;
const final = false;
const name = "One";
const value = null;
let item; // undefined
const id = new Symbol(1500);
```

◀ Can’t Use Proxy, not an ‘Object’

```
const invoice = {
  number
};
const items = [ invoice ];
const saleDate = new Date();
const aSet = new Set();
const aMap = new Map();
```

◀ Can use a proxy, they are ‘Object’s

Demo

What Can Be Wrapped by a Proxy

Traps

- Just object-level middleware
- Allows you to opt-into taking responsibility
- You can trap operations, not properties

Use-Cases

- Validation of Objects
- Notification of property changes
- Auditing
- Often used for reactivity

```
const person = {
  name: "Shawn"
}

const handler = {
  get: (target, prop) => {
    const val = target[prop];
    if (typeof val === "string") {
      return val.toUpperCase();
    }
    return val;
  }
};


const proxy =
  new Proxy(person, handler);



out(proxy.name);
```

- Handler is just a set of traps
- Your Custom Code
- Pass in traps when you create the proxy
- Trap is called when the property is accessed

Demo

Traps

```
function out(msg) {
  console.log(msg);
}

const handler = {
  apply: (target, thisArg, args) => {
    target.apply(thisArg, args);
  }
};



const proxy =
  new Proxy(out, handler);

proxy("hello");
```

- **apply** is trap for calling a function

- **apply** is called upon execution the proxy

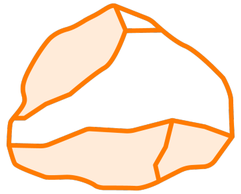
Demo

Trapping Functions

What We've Learned

Proxies are just surrogates for objects

Traps are just object-level middleware for operations



You can proxy anything that is an 'object'

Up Next:

Iteration with Proxies