Summary

Interfaces and type checking

Duck typing

Declare and use interfaces

Extend interfaces

Implement interfaces with classes

Demo

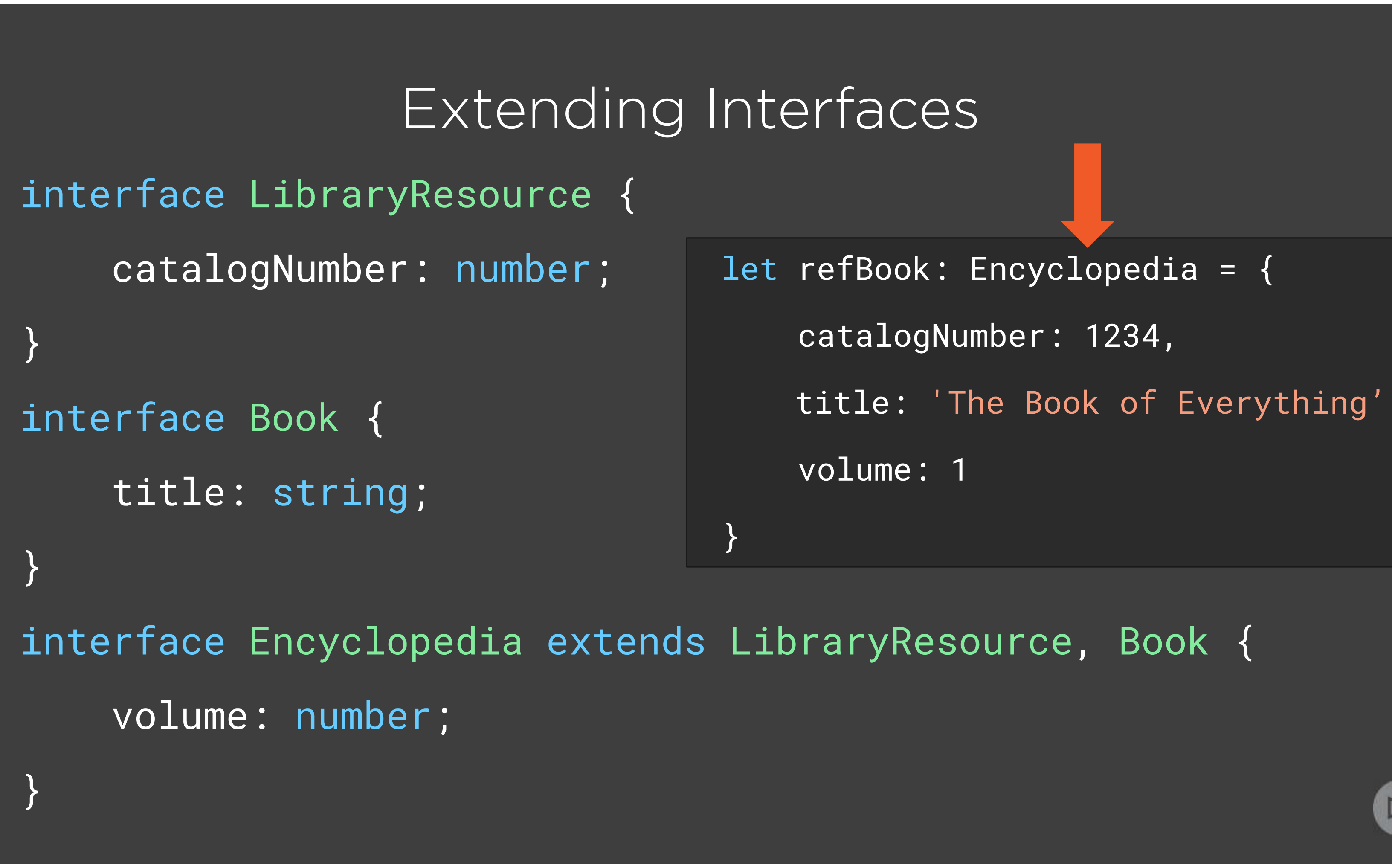
Implementing interfaces with classes

Class Types

```
interface Librarian {
    doWork: () => void;
}
class ElementarySchoolLibrarian implements Librarian {
    doWork() {
        console.log('Reading to and teaching children...');
    }
}
let kidsLibrarian: Librarian = new ElementarySchoolLibrarian();
kidsLibrarian.doWork();
```

Demo

Extending interfaces



Extending Interfaces

```
interface LibraryResource {
    catalogNumber: number;
}
interface Book {
    title: string;
}
interface Encyclopedia extends LibraryResource, Book {
    volume: number;
}
```

```
let refBook: Encyclopedia = {
    catalogNumber: 1234,
    title: 'The Book of Everything'
    volume: 1
}
```

Extending Interfaces

```
interface LibraryResource {
    catalogNumber: number;
}
interface Book {
    title: string;
}
interface Encyclopedia extends LibraryResource, Book {
    volume: number;
}
```

Extending Interfaces

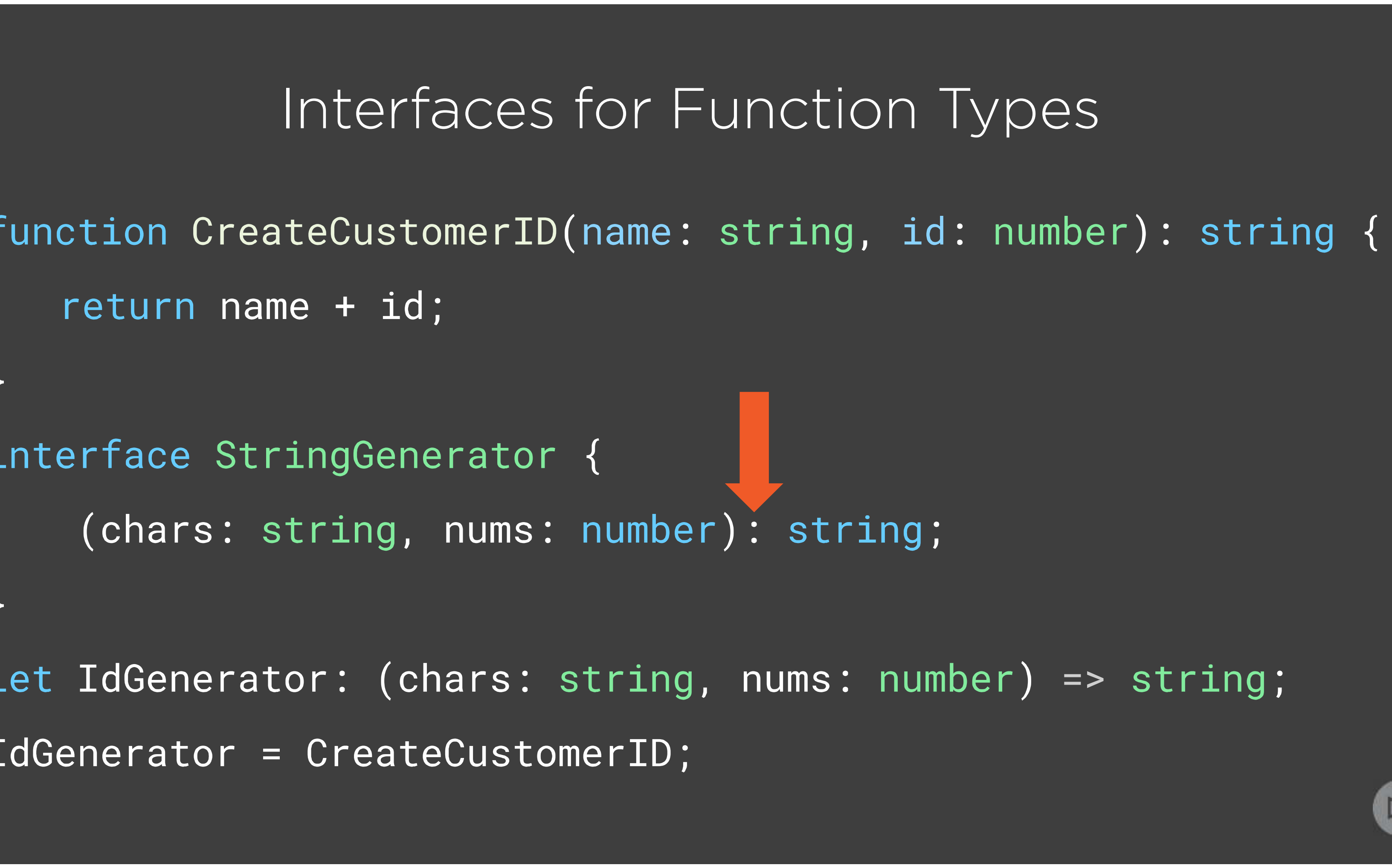
```
interface LibraryResource {
    catalogNumber: number;
}
interface Book {
    title: string;
}
interface Encyclopedia extends LibraryResource, Book {
    volume: number;
}
```

Demo

Interfaces for function types

Interfaces for Function Types

```
function CreateCustomerID(name: string, id: number): string {
   return name + id;
}
interface StringGenerator {
    (chars: string, nums: number): string;
}
let IdGenerator: StringGenerator;
IdGenerator = CreateCustomerID;
```



Interfaces for Function Types

```
function CreateCustomerID(name: string, id: number): string {
   return name + id;
}
nterface StringGenerator {
    (chars: string, nums: number): string;
}
et IdGenerator: (chars: string, nums: number) => string;
dGenerator = CreateCustomerID;
```

Interfaces for Function Types

```
function CreateCustomerID(name: string, id: number): string {
   return name + id;
}
interface StringGenerator {
    (chars: string, nums: number): string;
}
let IdGenerator: (chars: string, nums: number) => string;
IdGenerator = CreateCustomerID;
```

Demo

Defining and using interfaces

Demo

Restructuring the LibraryManager App

```
interface Book {
    id: number;
    title: string;
    author: string;
    pages?: number;
    markDamaged: (reason: string) => void;
}
```

Defining an Interface

“interface” keyword

List properties with their types

Optional properties denoted with “?”

Provide function signatures – no implementation

```
interface Book {
    id: number;
    title: string;
    author: string;
    pages?: number;

}
```

Defining an Interface

“interface” keyword

List properties with their types

Optional properties denoted with “?”

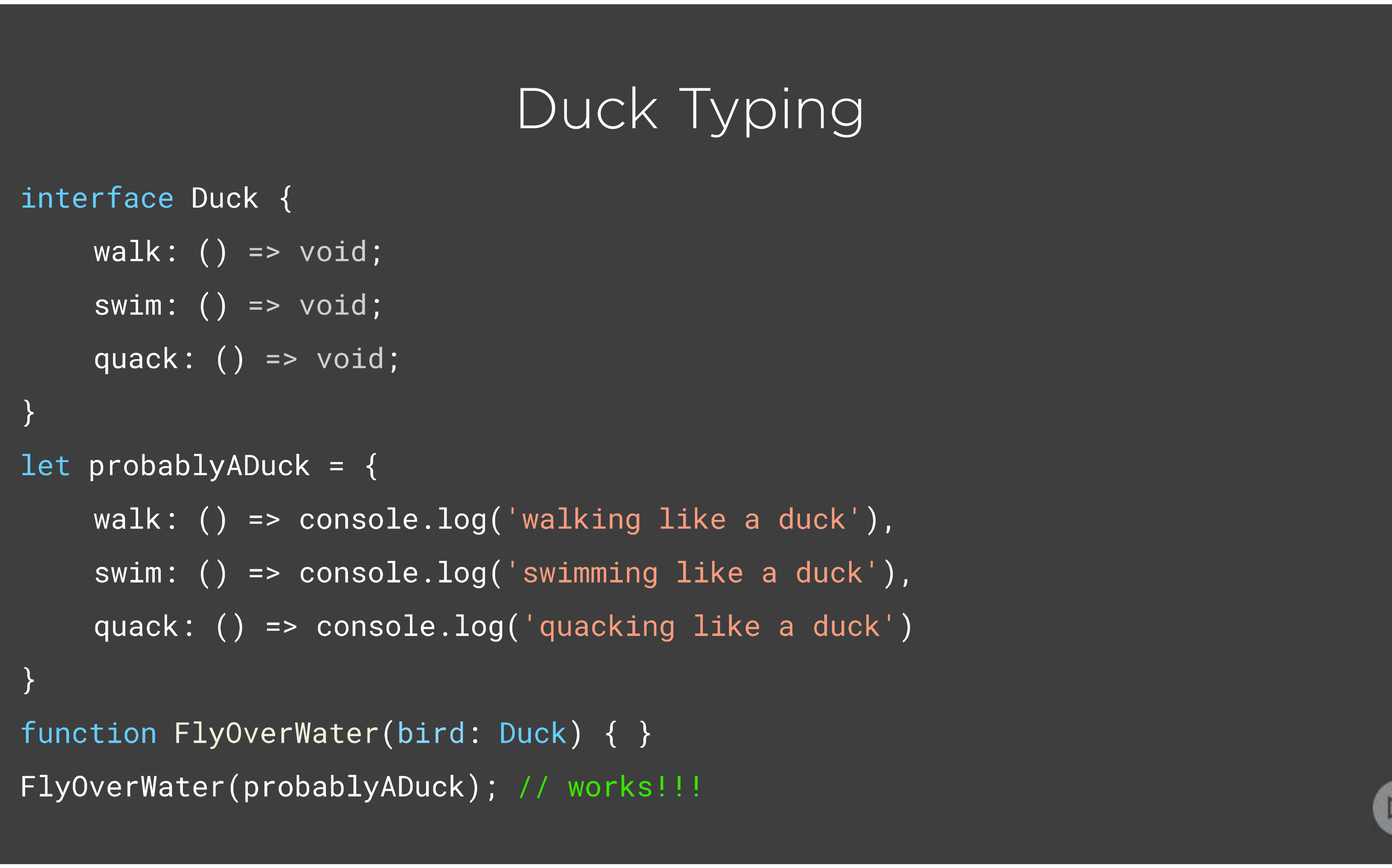
```
interface Book {
    id: number;
    title: string;
    author: string;
    pages?: number;

}
```

Defining an Interface

“interface” keyword

List properties with their types



Duck Typing

```
interface Duck {
    walk: () => void;
    swim: () => void;
    quack: () => void;
}
let probablyADuck = {
    walk: () => console.log('walking like a duck'),
    swim: () => console.log('swimming like a duck'),
    quack: () => console.log('quacking like a duck')
}
function FlyOverWater(bird: Duck) { }
FlyOverWater(probablyADuck); // works!!!
```

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.”

James Whitcomb Riley

Source: https://en.wikipedia.org/wiki/Duck_test

Contracts that define types

Compiler enforces the contract via type checking

Collection of property and method definitions

Duck typing

Overview

What is an interface?

Duck typing

Declare interfaces

Interfaces for function types

Extending interfaces

Interfaces for class types