

INNOQ TECHNOLOGY LUNCH / 19.10.2022

# Advanced Typing in TypeScript

**INNOQ**



**LARS HUPEL**

@larsr\_h · they/them

# TypeScript is JavaScript with syntax for types.

TypeScript is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.

Try TypeScript Now  
Online or via npm



Editor Checks Auto-complete Interfaces JSX

```
const user = {  
  firstName: "Angela",  
  lastName: "Davis",  
  role: "Professor",  
}
```

```
console.log(user.name)
```

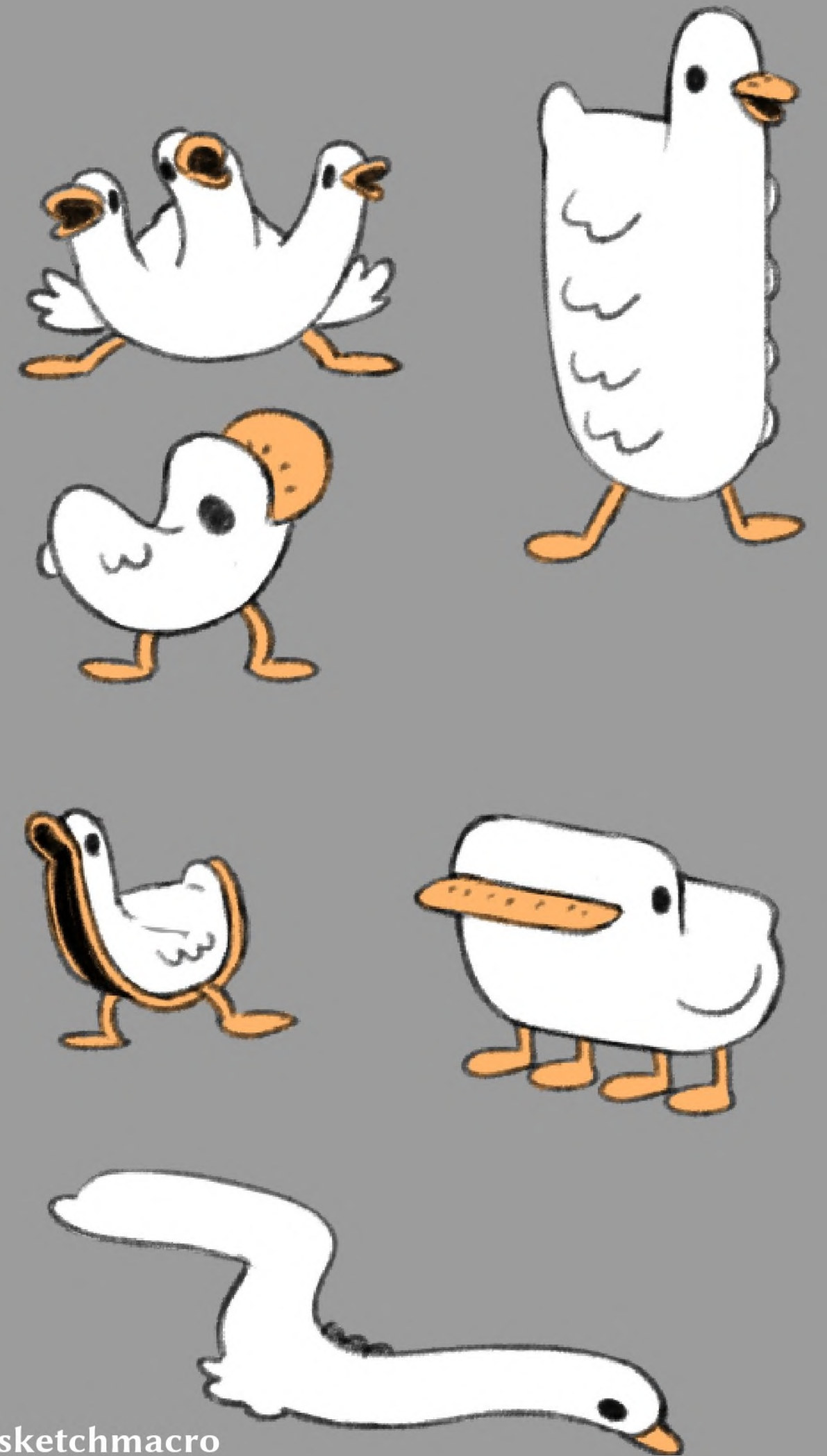
Property 'name' does not exist on type '{ firstName: string; lastName: string; role: string; }'.



<https://www.typescriptlang.org/>

# Core tenets

- JavaScript + X
- Environment independence
- Structural typing (aka "duck typing")
- Type inference
- Support of wonky JavaScript patterns



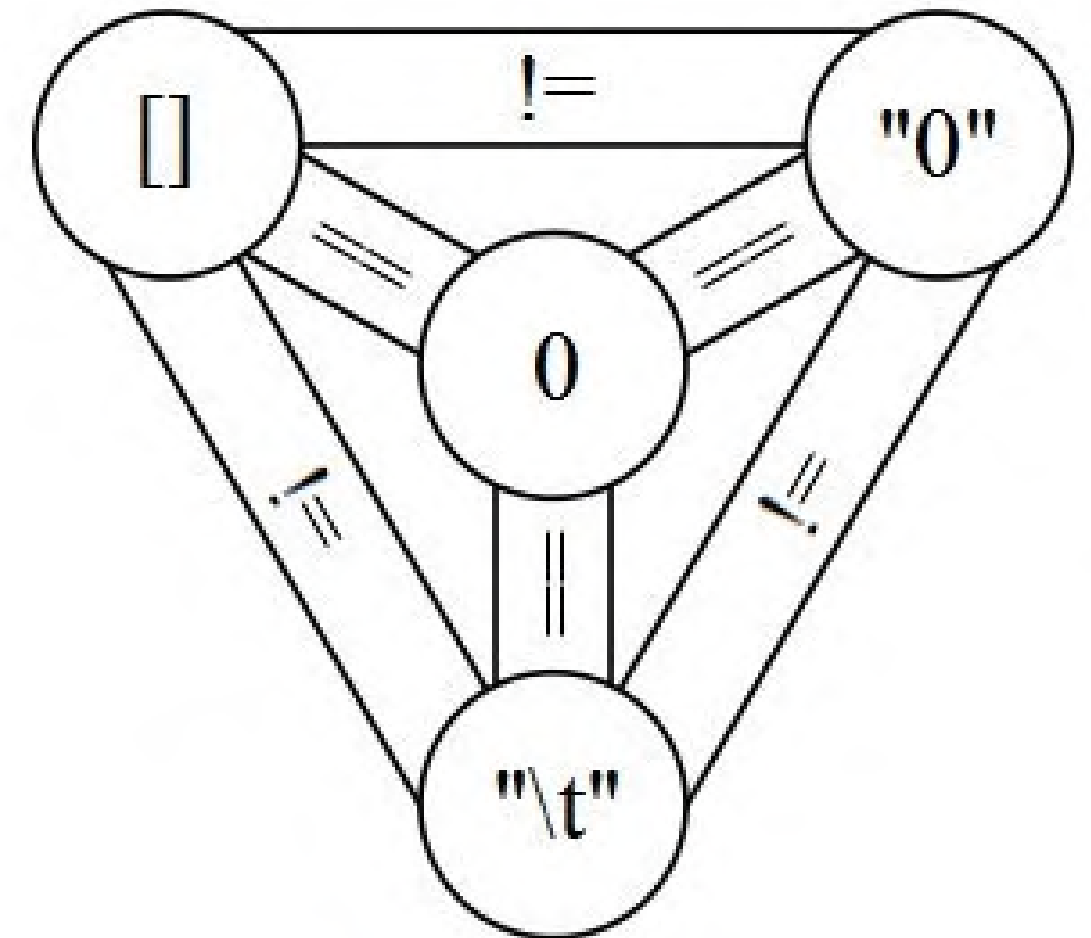
The background of the slide is a dense, abstract composition of numerous thin, translucent, fiber-like structures. These fibers are oriented in various directions, creating a complex, web-like pattern. The fibers exhibit a range of colors, including shades of blue, purple, pink, and orange, which are most prominent at their tips and along their lengths. The overall effect is one of dynamic movement and intricate detail, set against a dark, almost black, background.

# **Chapter 1**

**What is JS?**

# JS does not stop you from ...

- adding arrays to objects
- multiplying objects with numbers
- comparing 🍏 with 🍊



JavaScript

@hsjoihs

**TypeScript set out to fix this.**

**... and fixing they did**

**What JS devs  
asked for**



**What JS devs  
got**



**"Types are a sign of  
defeat. You lost  
control of your life  
and then you just  
use types to  
program  
JavaScript."**



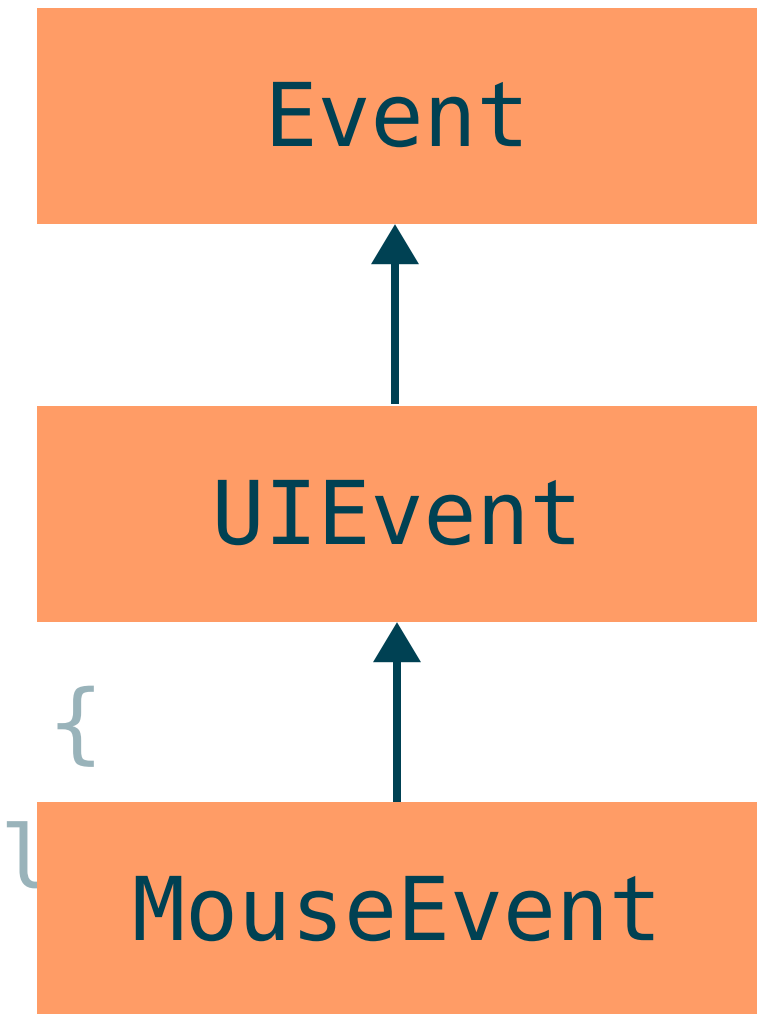
# Chapter 2

## Typing the DOM

```
addEventListener(type, listener);  
addEventListener(type, listener, options);  
addEventListener(type, listener, useCapture);
```

```
button.addEventListener( 'click', event => {  
    console.log( `Click count: ${event.detail}` );  
});
```

```
button.addEventListener('click', event => {  
  console.log(`Click count: ${event.detail}`);  
});
```



```
interface GlobalEventHandlersEventMap {  
  "abort": UIEvent;  
  "animationcancel": AnimationEvent;  
  // ...  
  "click": MouseEvent;  
  // ...  
}
```

```
interface GlobalEventHandlers {  
  addEventListener<K extends keyof GlobalEventHandlersEventMap>(  
    type: K,  
    listener: (ev: GlobalEventHandlersEventMap[K]) => any,  
    options?: boolean | AddEventListenerOptions  
  ): void;  
}
```

```

interface GlobalEventHandlersEventMap {
    "abort": UIEvent;
    "animationcancel": AnimationEvent;
    // ...
    "click": MouseEvent;
    // ...
}

```

```

"abort" | "animationcancel" | "click" | /* ... */

```

```

interface GlobalEventHandlers {
    addEventListener<K extends keyof GlobalEventHandlersEventMap>(
        type: K,
        listener: (ev: GlobalEventHandlersEventMap[K]) => any,
        options?: boolean | AddEventListenerOptions
    ): void
}

```

```

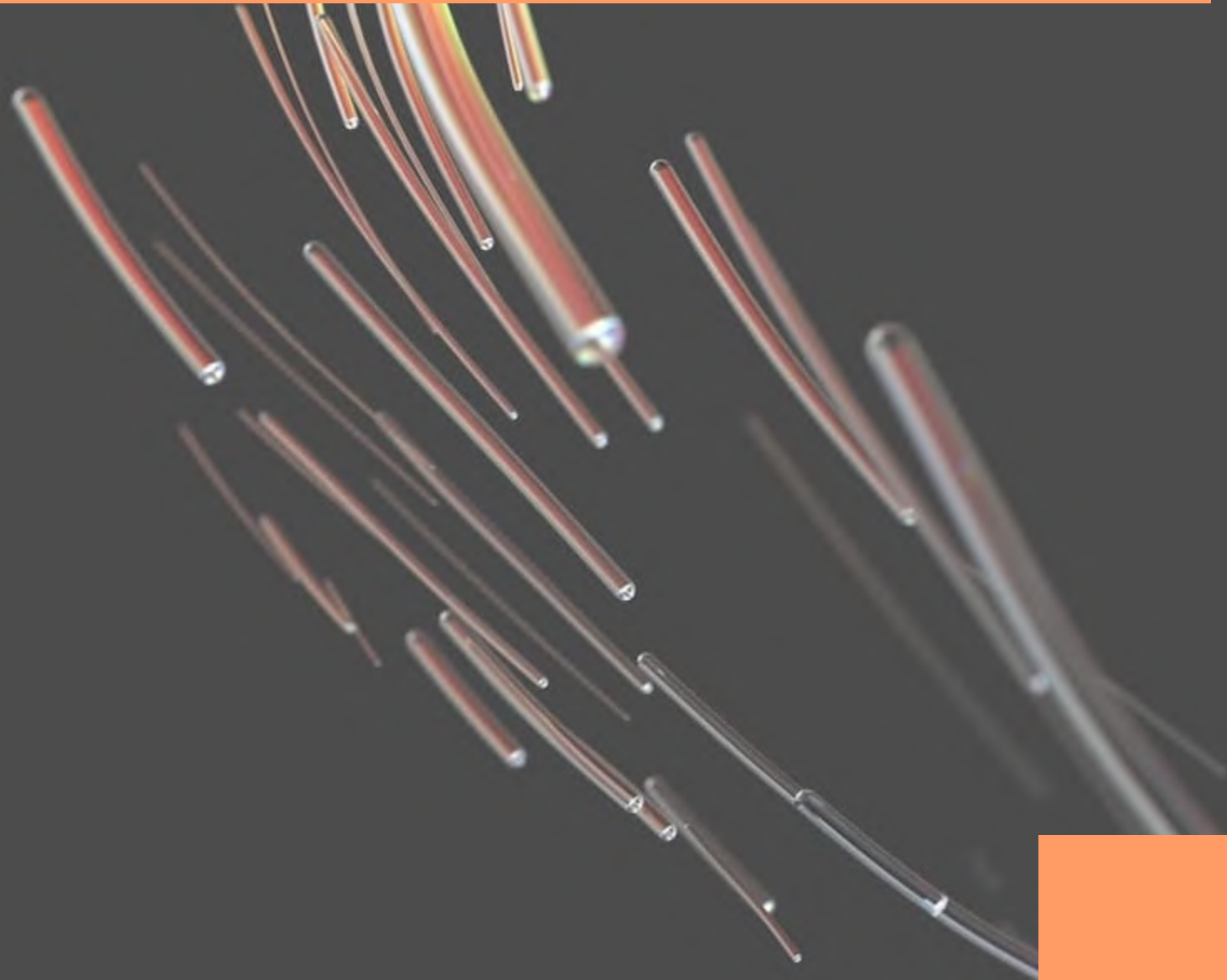
GlobalEventHandlersEventMap["click"]

```

**same thing for Node ...**



# Chapter 3



**Dataframes!** 

# Data analysis

- Data Science™ is basically just tables, right?
- CSV is a great format for tables
- ... but what do we do with them?



# Typical operations

- access row  $i$  at column  $c$
- filter rows by predicate
- drop columns
- add new column
- aggregate & group by column



```
operation(table: Table): Table
```

```
operation(table: Table): Table
```



```
operation(table: Table<ColsIn>): Table<ColsOut>
```

```
operation(table: Table<ColsIn>): Table<ColsOut>
```



```
operation(table: Table<ColsIn>): Table<ColsOut>
```



```
interface Table<  
    Cols extends Record<string, any>  
>
```

| # | Order ID | Item  | Qty | Price |
|---|----------|-------|-----|-------|
| 1 | ABC      | Apple | 5   | 10    |
| 2 | ABC      | Pear  | 2   | 3     |
| 3 | DEF      | Apple | 1   | 8     |

| # | Order ID | Item  | Qty | Price |
|---|----------|-------|-----|-------|
| 1 | ABC      | Apple | 5   | 10    |
| 2 | ABC      | Pear  | 2   | 3     |
| 3 | DEF      | Apple | 1   | 8     |

Table<{

orderId: string;

item: string;

qty: number;

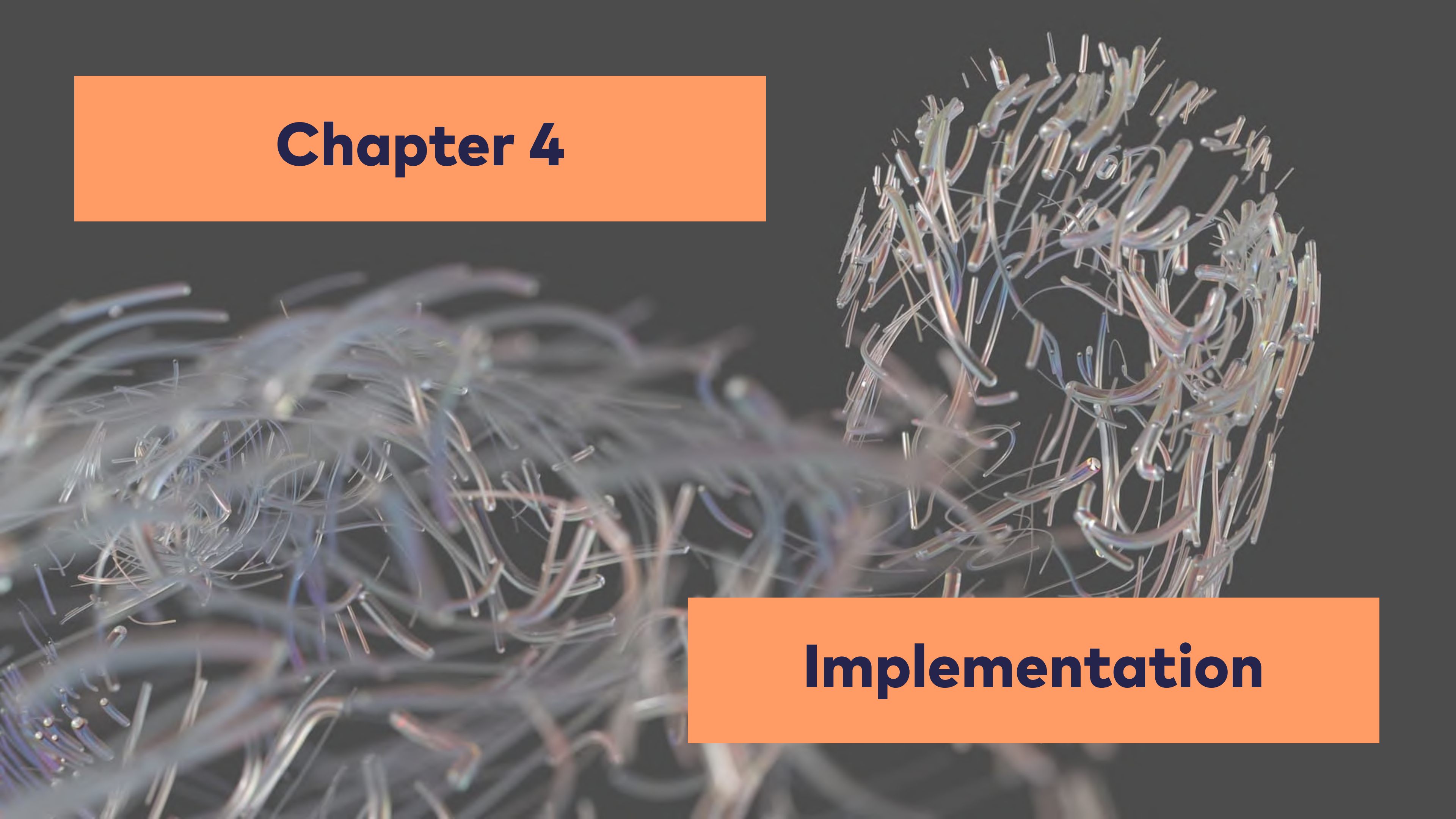
price: number;

}>



# Chapter 4

## Implementation



```
interface Series<Elem> {  
    get length(): number;  
    at(n: number): Elem;  
}
```

```
interface Table<Index, Cols extends Record<string, any>> {  
  get index(): Series<Index>  
  select<Col extends keyof Cols>(col: Col): Series<Cols[Col]>  
  lookup(index: Index): Row<Index, Cols> | undefined  
}
```



**Demo**

# Q&A



Lars Hupel (they/them)  
lars.hupel@innoq.com  
@larsr\_h

## **innoQ Deutschland GmbH**

Krischerstr. 100  
40789 Monheim  
+49 2173 333660

Ohlauer Str. 43  
10999 Berlin

Ludwigstr. 180E  
63067 Offenbach

Kreuzstr. 16  
80331 München

Hermannstr. 13  
20095 Hamburg

Erftstr. 15-17  
50672 Köln

Königstorggraben 11  
90402 Nürnberg

# Images

- Karl Lagerfeld by Adach, CC-BY-SA 2.0,  
[https://commons.wikimedia.org/w/index.php?title=File:Karl\\_Lagerfeld\\_\(14091153382\)\\_\(cropped\).jpg&oldid=477165629](https://commons.wikimedia.org/w/index.php?title=File:Karl_Lagerfeld_(14091153382)_(cropped).jpg&oldid=477165629)
- Mechanical adding machine by Nxr-at, CC-BY-SA 4.0,  
[https://commons.wikimedia.org/w/index.php?title=File:Mechanical\\_adding\\_machines\\_WW\\_Continental.JPG&oldid=470727928](https://commons.wikimedia.org/w/index.php?title=File:Mechanical_adding_machines_WW_Continental.JPG&oldid=470727928)