TYPESCRIPT

TOPICS FOR JAVASCRIPT PROGRAMMERS

Prof. Andrew Sheehan

Boston University/MET
Computer Science Department

MORE BACKGROUND ON MODULES

Modules began with JavaScript in 2012.

ES6 Modules became a core standard in 2015.

Every modern browser uses modules since 2020.

In TypeScript, just as in ECMAScript 2015, any file containing a top-level import or export is considered a module.

INFERENCE

Explanation 1: *Type inference is the ability to automatically deduce the type of an expression at compile time.*

Explanation 2: *Type inference is the process of reconstructing missing type information in a program based on the usage of its variables.*

```
let account = {
    customer: "Default (Unknown)",
    id: 0
}
```

Based on above, the type structure is:

```
let account = {
    customer: string,
    id: number
}
```

```
let works:Account = {customer: "Hayes", id:1}
Let doesNotWork:Account = {sku: "12za2", id:2}
```

'sku' does not exist on Account type.

INFERENCE WITH TYPES

CONFORMING TO THE SHAPE

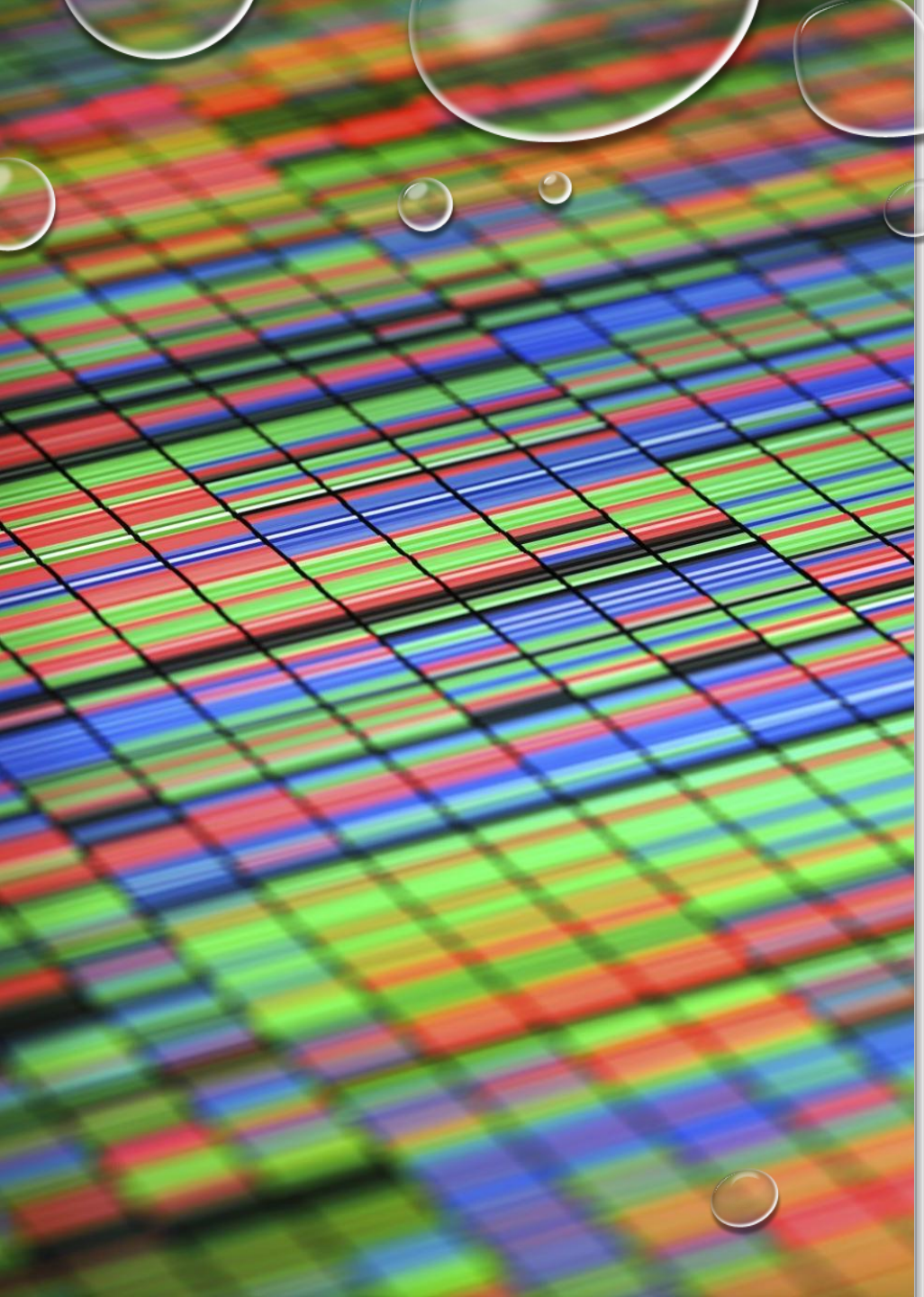
If your type and use match the interface, it will work.

```
interface User {
  name: string;
  id: number;
}

class UserAccount {
  name: string;
  id: number;

  constructor(name: string, id: number) {
    this.name = name;
    this.id = id;
  }
}

const user: User = new UserAccount("Murphy", 1);
```

TYPEOF

To learn the type of a variable, use typeof

Type	Predicate
string	typeof s === 'string'
number	typeof n === 'number'
boolean	typeof b === 'boolean'
undefined	typeof undefined === 'undefined'
function	typeof f === 'function'
array	Array.isArray(a)



GENERICS

They provide a way to tell functions, classes, or interfaces what type you want to use when you call it.

```
let lotteryNumbers = new List<number>();
lotteryNumbers.add(20);
lotteryNumbers.add(55);
```

```
let babyNames = new List<string>();
babyNames.add("Francis");
babyNames.add("Helen");
```

The shape of a thing determines type. Called ‘duck typing’ or ‘structural typing’

If 2 objects have the same shape, they are of the same type.

```ts
interface Point {
  x: number;
  y: number;
}

function logPoint(p: Point) {
  console.log(`${p.x}, ${p.y}`);
}

// logs "12, 26"
const point = { x: 12, y: 26 };
logPoint(point);
```

STRUCTURAL
TYPE SYSTEM

NAMESPACE

TypeScript has its own module format called namespace which pre-dates the ES Modules standard.

Do not use it. Use modules

```
namespace Validation {
  export interface StringValidator {
    isAcceptable(s: string): boolean;
  }
}
/// <reference path="Validation.ts" />
namespace Validation {
  const lettersRegexp = /^[A-Za-z]+$/;
  export class LettersOnlyValidator implements StringValidator {
    isAcceptable(s: string) {
      return lettersRegexp.test(s);
    }
  }
}
```

TYPESCRIPT MIXINS

Mixins are special classes that contain a combination of methods that can be used by other classes

They promote code reusability & help you avoid limitations associated with multiple inheritance.

blog.logrocket.com/typescript-mixins-examples-and-use-cases