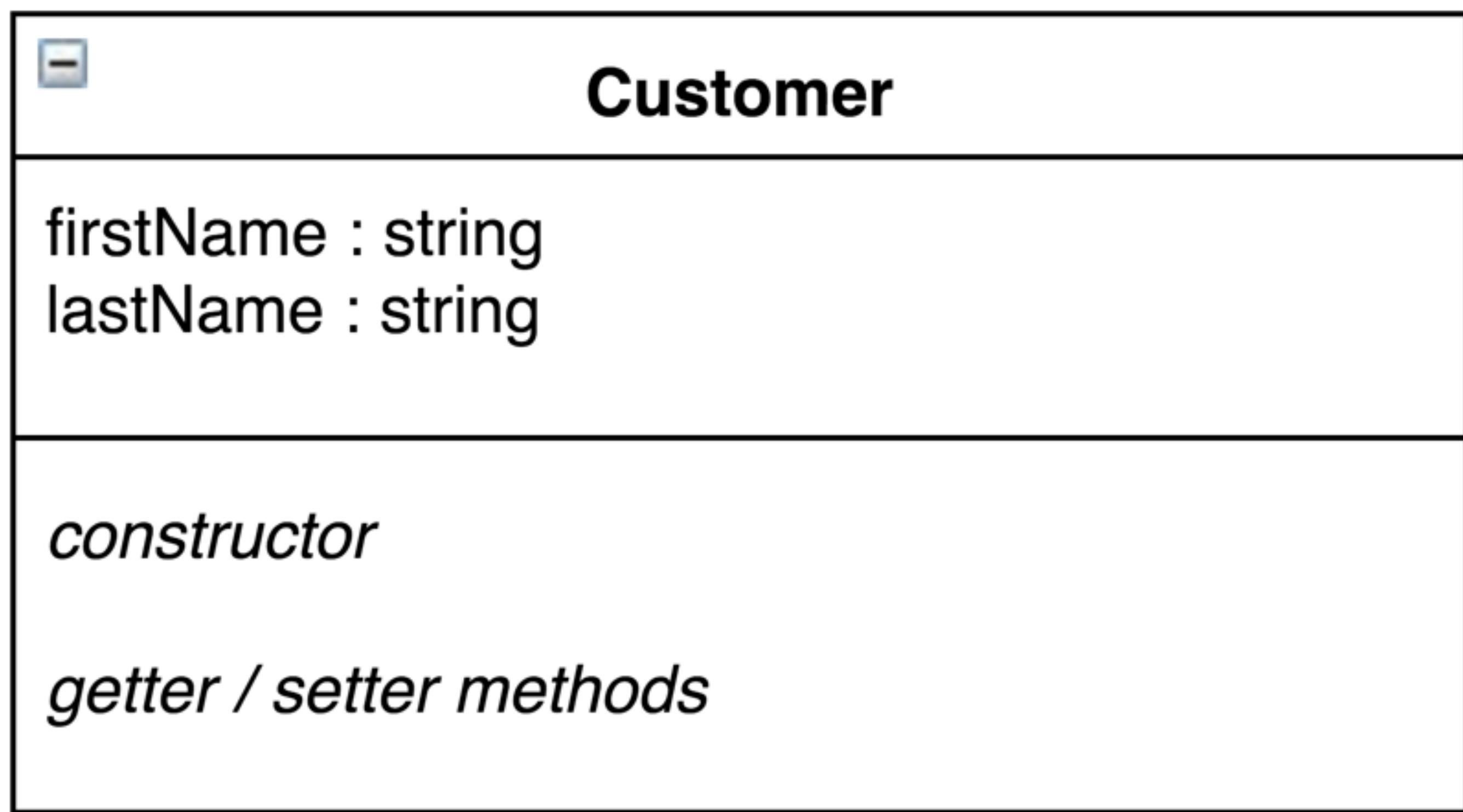


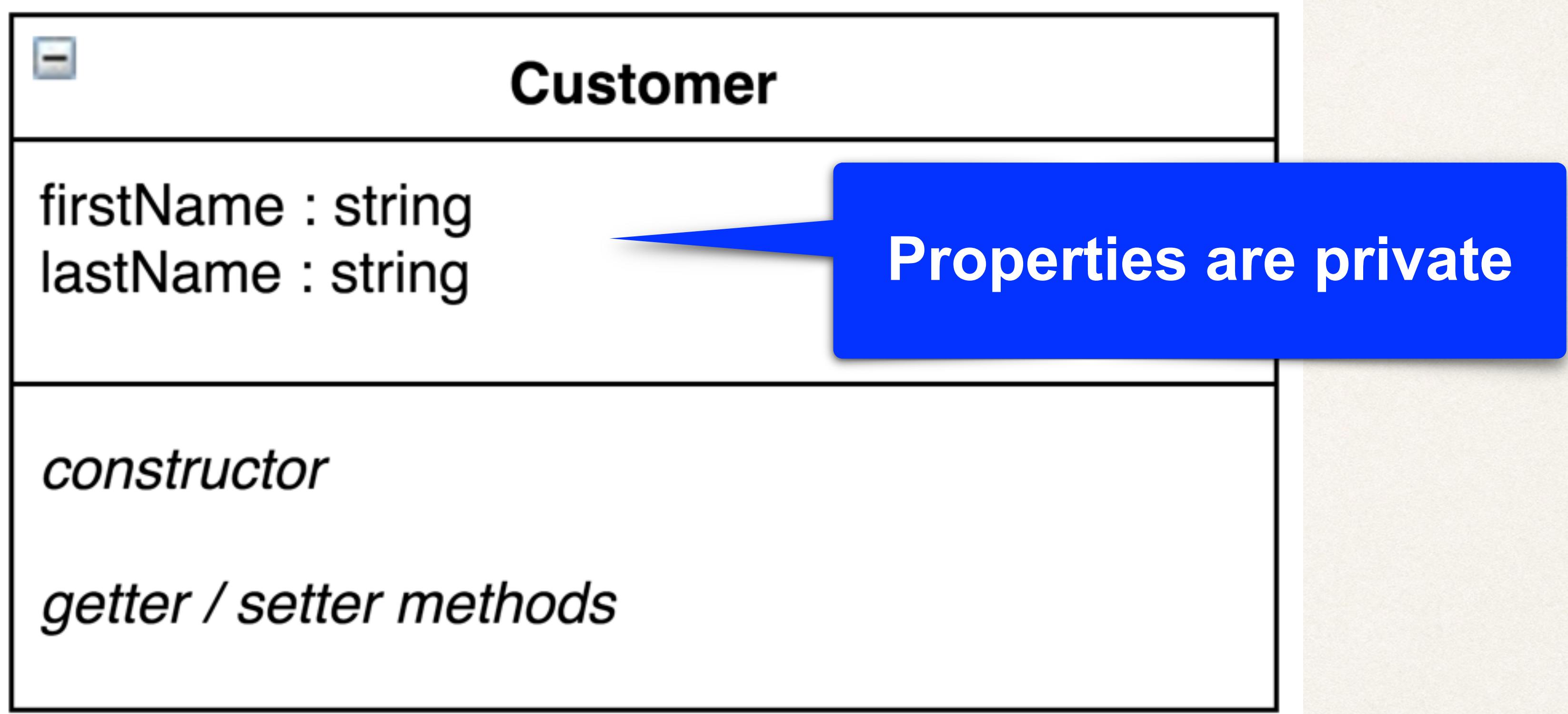
TypeScript - Accessors



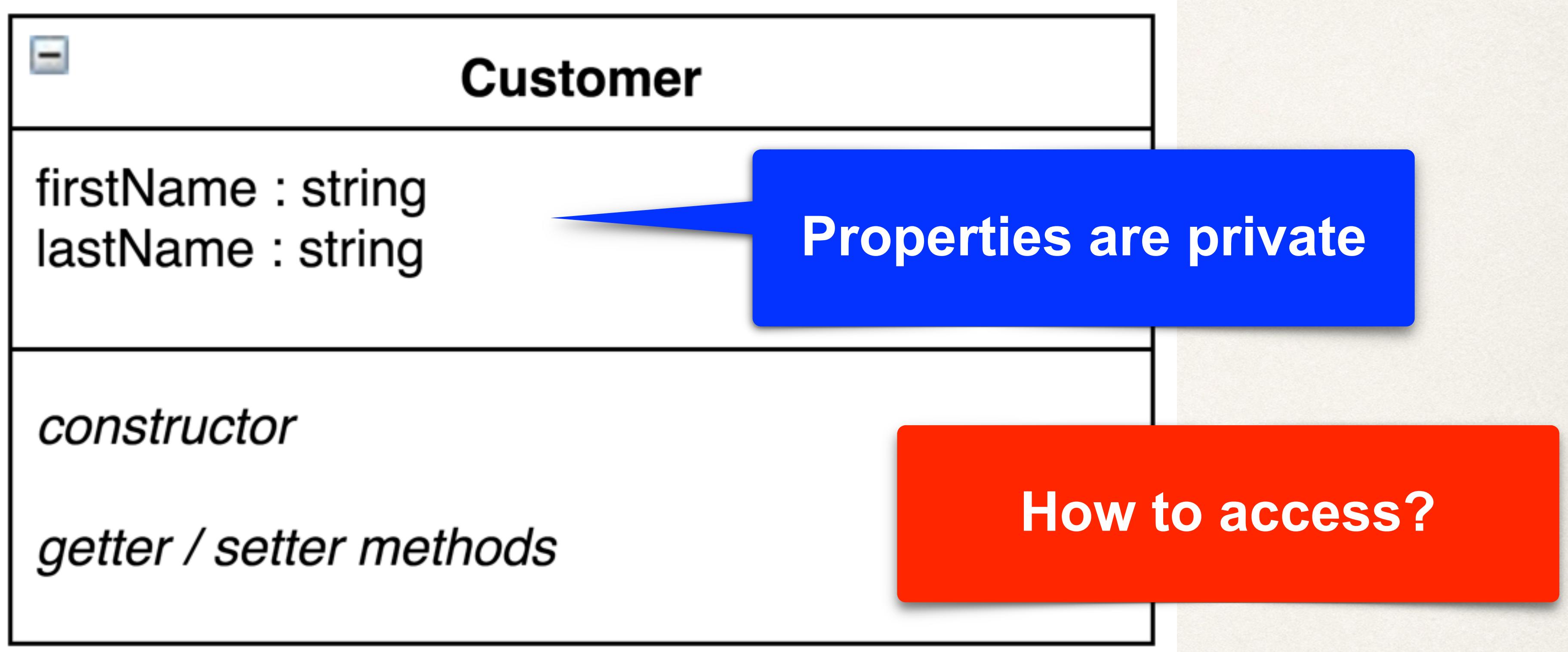
Customer Class



Customer Class



Customer Class



Getter / Setter Methods

Getter / Setter Methods

- Since our properties are private, we need a way to access them

Getter / Setter Methods

- Since our properties are private, we need a way to access them
- We can create traditional methods as in other OO languages:

Getter / Setter Methods

- Since our properties are private, we need a way to access them
- We can create traditional methods as in other OO languages:
 - Define getter / setter methods

Getter / Setter Methods

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
    ... ...
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
    ... ...  
  
    public getFirstName(): string {  
        return this.firstName;  
    }  
}
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
    Method name  
  
    private firstName: string;  
    private lastName: string;  
    ...  
  
    public getFirstName(): string {  
        return this.firstName;  
    }  
}
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
    private firstName: string;  
    private lastName: string;  
    ...  
  
    public getFirstName(): string {  
        return this.firstName;  
    }  
}
```

Method name

Return type

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
    ... ...  
  
    public getFirstName(): string {  
        return this.firstName;  
    }  
  
    public setFirstName(theFirst: string): void {  
        this.firstName = theFirst;  
    }  
}
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
  
    Method name  
    public get firstName(): string {  
        return this.firstName;  
    }  
  
    public setfirstName(theFirst: string): void {  
        this.firstName = theFirst;  
    }  
}
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
  
    Method name           Param type  
    public get firstName(): string {  
        return this.firstName;  
    }  
  
    public set firstName(theFirst: string): void {  
        this.firstName = theFirst;  
    }  
}
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
  
    Method name  
    public get firstName(): string {  
        return this.firstName;  
    }  
  
    public setfirstName(theFirst: string): void {  
        this.firstName = theFirst;  
    }  
}
```

Param type

Return type

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
    ... ...  
  
    public getFirstName(): string {  
        return this.firstName;  
    }  
  
    public setFirstName(theFirst: string): void {  
        this.firstName = theFirst;  
    }  
}
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
    ... ...  
  
    public getFirstName(): string {  
        return this.firstName;  
    }  
  
    public setFirstName(theFirst: string): void {  
        this.firstName = theFirst;  
    }  
}
```

```
// now let's use it  
let myCustomer = new Customer("Martin", "Dixon");
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
    ... ...  
  
    public getFirstName(): string {  
        return this.firstName;  
    }  
  
    public setFirstName(theFirst: string): void {  
        this.firstName = theFirst;  
    }  
}
```

```
// now let's use it  
let myCustomer = new Customer("Martin", "Dixon");  
  
myCustomer.setFirstName("Greg");
```

Getter / Setter Methods

File: Customer.ts

```
class Customer {  
  
    private firstName: string;  
    private lastName: string;  
    ... ...  
  
    public getFirstName(): string {  
        return this.firstName;  
    }  
  
    public setFirstName(theFirst: string): void {  
        this.firstName = theFirst;  
    }  
}
```

```
// now let's use it  
let myCustomer = new Customer("Martin", "Dixon");  
  
myCustomer.setFirstName("Greg");  
console.log(myCustomer.getFirstName());
```

TypeScript: Accessors - Get / Set

TypeScript: Accessors - Get / Set

- TypeScript also offers an alternate syntax

TypeScript: Accessors - Get / Set

- TypeScript also offers an alternate syntax
- Define special: get / set methods

TypeScript: Accessors - Get / Set

- TypeScript also offers an alternate syntax
- Define special: get / set methods
- Known as **Accessors**

TypeScript: Accessors - Get / Set

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
  
    ... ...
```

Can give any property name
Some TypeScript developers use leading "_"

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
  
    ... ...
```

Can give any property name
Some TypeScript developers use leading "_"

Nothing special about "_"
No special magic
Just a convention

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ...  
  
    public get firstName(): string {  
        return this._firstName;  
    }  
}
```

Note the syntax:
get <space> property()

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ...  
  
    public get firstName(): string {  
        return this._firstName;  
    }  
  
    public set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

Note the syntax:
set <space> property(...)

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ...  
  
    public get firstName(): string {  
        return this._firstName;  
    }  
  
    public set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

Note the syntax:
set <space> property(...)

No return type ... not even "void"

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ... ...  
  
    public get firstName(): string {  
        return this._firstName;  
    }  
  
    public set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

```
// now let's use it  
let myCustomer = new Customer("Martin", "Dixon");
```

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ...  
  
    public get firstName(): string {  
        return this._firstName;  
    }  
  
    public set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

```
// now let's use it  
let myCustomer = new Customer("Martin", "Dixon");  
  
myCustomer.firstName = "Susan";
```

Calls the "set" accessor
since we are
assigning a value

TypeScript: Accessors - Get / Set

Calls the "get" accessor
since we are
accessing a value

```
private _firstName: string;
private _lastName: string;
...
public get firstName(): string {
    return this._firstName;
}

public set firstName(value: string) {
    this._firstName = value;
}
```

```
// now let's use it
let myCustomer = new Customer("Martin", "Dixon");

myCustomer.firstName = "Susan";
console.log(myCustomer.firstName);
```

Calls the "set" accessor
since we are
assigning a value

TypeScript: Accessors - Getters and Setters

Calls the "get" accessor
since we are
accessing a value

```
private _firstName: string;
private _lastName: string;
...
public get firstName(): string {
    return this._firstName;
}

public set firstName(value: string) {
    this._firstName = value;
}
```

```
// now let's use it
let myCustomer = new Customer("Martin", "Dixon");

myCustomer.firstName = "Susan";
console.log(myCustomer.firstName);
```

Code outside of the class is NOT
accessing internal properties directly

Calls the "set" accessor
since we are
assigning a value

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private x: string;  
    private y: string;  
    ... ...  
  
    public get firstName(): string {  
        return this.x;  
    }  
  
    public set firstName(value: string) {  
        this.x = value;  
    }  
}
```

```
// now let's use it  
let myCustomer = new Customer("Martin", "Dixon");  
  
myCustomer.firstName = "Susan";  
console.log(myCustomer.firstName);
```

TypeScript: Accessors - Get / Set

Can give any internal name

File: Customer.ts

```
class Customer {  
  
    private x: string;  
    private y: string;  
    ... ...  
  
    public get firstName(): string {  
        return this.x;  
    }  
  
    public set firstName(value: string) {  
        this.x = value;  
    }  
}
```

```
// now let's use it  
let myCustomer = new Customer("Martin", "Dixon");  
  
myCustomer.firstName = "Susan";  
console.log(myCustomer.firstName);
```

TypeScript: Accessors - Get / Set

Can give any internal name

File: Customer.ts

```
class Customer {  
  
    private x: string;  
    private y: string;  
    ... ...  
  
    public get firstName(): string {  
        return this.x;  
    }  
  
    public set firstName(value: string) {  
        this.x = value;  
    }  
}
```

```
// now let's use it  
let myCustomer = new Customer("Martin", "Dixon");  
  
myCustomer.firstName = "Susan";  
console.log(myCustomer.firstName);
```

The public get/set accessors
are still called accordingly

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ... ...  
  
    get firstName(): string {  
        return this._firstName;  
    }  
  
    set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ... ...  
  
    get firstName(): string {  
        return this._firstName;  
    }  
  
    set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

Renamed to the original names
from previous slides

TypeScript: Accessors - Get / Set

Removed “public” on the accessors.

File: Cust

If no access modifier given, “public” by default

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ...  
  
    get firstName(): string {  
        return this._firstName;  
    }  
  
    set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

Compiler flag

Compiler flag

- The get/set accessors feature is only supported in ES5 and higher

Compiler flag

- The get/set accessors feature is only supported in ES5 and higher
- You have to set a compiler flag in order to compile the code

Compiler flag

- The get/set accessors feature is only supported in ES5 and higher
- You have to set a compiler flag in order to compile the code

```
C:\> tsc --target ES5 --noEmitOnError Customer.ts
```

Compiler flag

Problem with too many compiler flag

Problem with too many compiler flag

- You may have noticed, that we have a lot of compiler flags

Problem with too many compiler flag

- You may have noticed, that we have a lot of compiler flags
 - Too much stuff to remember ... easy to forget

Problem with too many compiler flag

- You may have noticed, that we have a lot of compiler flags
 - Too much stuff to remember ... easy to forget
- Wouldn't it be great to set this up in a config file?

Problem with too many compiler flag

- You may have noticed, that we have a lot of compiler flags
 - Too much stuff to remember ... easy to forget
- Wouldn't it be great to set this up in a config file?
- TypeScript has a solution: **tsconfig.json** file

tsconfig.json

tsconfig.json

- **tsconfig.json** file defines compiler options and project settings

tsconfig.json

- **tsconfig.json** file defines compiler options and project settings
- Place this file in the root of your project directory

tsconfig.json

- **tsconfig.json** file defines compiler options and project settings
- Place this file in the root of your project directory

File: tsconfig.json

```
{  
  "compilerOptions": {  
    "noEmitOnError": true,  
    "target": "es5"  
  }  
}
```

tsconfig.json

tsconfig.json

- You can also generate a template for this file

tsconfig.json

- You can also generate a template for this file

```
C:\> tsc --init
```

tsconfig.json

- You can also generate a template for this file



tsconfig.json

- You can also generate a template for this file



- Then edit the **tsconfig.json** accordingly for your project requirements

Compiling your Project

Compiling your Project

- Once your project has a **tsconfig.json** file, then you can compile with

Compiling your Project

- Once your project has a **tsconfig.json** file, then you can compile with

```
C:\> tsc
```

Compiling your Project

- Once your project has a **tsconfig.json** file, then you can compile with

```
C:\> tsc
```

No need to give names of TypeScript files.
By default, will compile all *.ts files

Compiling your Project

- Once your project has a **tsconfig.json** file, then you can compile with

```
C:\> tsc
```

No need to give names of TypeScript files.
By default, will compile all *.ts files

www.luv2code.com/tsconfig-docs

TypeScript: Accessors - Get / Set

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ... ...  
  
    get firstName(): string {  
        return this._firstName;  
    }  
  
    set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

TypeScript: Accessors - Get / Set

File: Customer.ts

```
class Customer {  
  
    private _firstName: string;  
    private _lastName: string;  
    ... ...  
  
    get firstName(): string {  
        return this._firstName;  
    }  
  
    set firstName(value: string) {  
        this._firstName = value;  
    }  
}
```

// now let's use it
let myCustomer = new Customer("Martin", "Dixon");

myCustomer.firstName = "Susan";
console.log(myCustomer.firstName);