Why you should consider using Typescript

By. Talha Rahman & Brian Cheung

Hunter College Major Capstone - Spring 2021

5/17/21

What is Typescript?

- Syntactical superset of Javascript with *optional* static typing
- Transpiles down to Javascript
- Made public in 2012
- Developed and maintained by Microsoft

Common Issues with Javascript

- Runtime errors due to unexpected values
  - Accessing invalid properties in objects
  - Function calls with unintended parameters
  - Javascript implicit type coercion
- Lackluster autocompletion & documentation
- Potentially unclear code

Javascript Issue - Accessing Invalid Property

- Possible to access non-defined properties of objects
  - Does not result in an Error
- Accessing nested properties can lead to a runtime Error

```
const person = {
    name: "J",
    age: 21
}

console.log(person.shame);
// undefined
console.log(person.id.toStr());
// TypeError: Cannot read
// property "toStr" of undefined
```

Javascript Issue - Invalid Function Usage

- Possible to call functions with
  - More or fewer parameters than originally intended
  - Incorrect data types
- Does not result in an Error
- Leads to unintended behavior

```
function add(num1, num2) {
    console.log(num1 + num2)
}

add("cat")
// catundefined
```

Javascript Type Coercion

- Javascript's goal is to keep running no matter what
- Type coercion converts unexpected values to "expected" values
  - Leads to unintended behavior
- Developers have to be aware of truthy/falsy values

```
1 + "2" + 1;
// 121
true + true
// 2
4 * []
// 0
4 * [2]
// 8
4 + [2]
// 42
```

```
NaN === NaN
// false
false == []
// true
isNaN("text")
// true
3 === new Number(3)
// false
3 === Number(3)
// true
```

Javascript Issue - Unclear Code

- Possible to write Javascript code that is valid, but the intention is unclear
  - Reasons: dynamic typing, type coercion or other weird features of Javascript
- Javascript code can have no errors and appear functional, but does not work as intended

```
function addNumAndSort(list, num) {
    list.push(num)
    list.sort()
}
```

- Is "list" an array? A user-defined datatype?
- is "num" a number, a float, a string?
- list.sort() converts values to strings and then sorts by string values.

Easing the burden of Javascript

- Extensive comments with details on types and how to use functions
- Developer time wasted
  - More time reading code
  - More time reading documentation
  - More time debugging issues
- Using workarounds for Types such as "proptypes" for React

Typescript in Action

- Typescript can infer types
- Prevents changing of types
- Adjustable strictness of type-checking
- Can opt-out

```
let str = "Hello World";

str = 5 //Type 'number' is not assignable to type 'string'
```

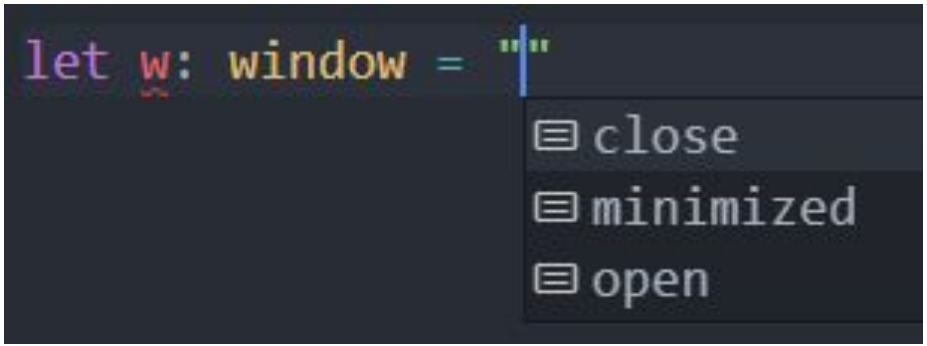
```
const list: Array<number> = [1, 2, 3, "str"]
// Type 'string' is not assignable to type 'number'
```

```
const list: Array<any> = [1, 2, 3, "str"]
```

Typescript Usage - User-defined Types

- Typescript allows for User-defined types
  - Can use expressions to allow for multiple types
- User-defined types allow for auto-completion

```
type window = "open" | "close" | "minimized"
```



Typescript Usage - Object Interfaces

- Define custom interfaces for objects
- Throws error if objects don't conform to interface
- Catches invalid property access

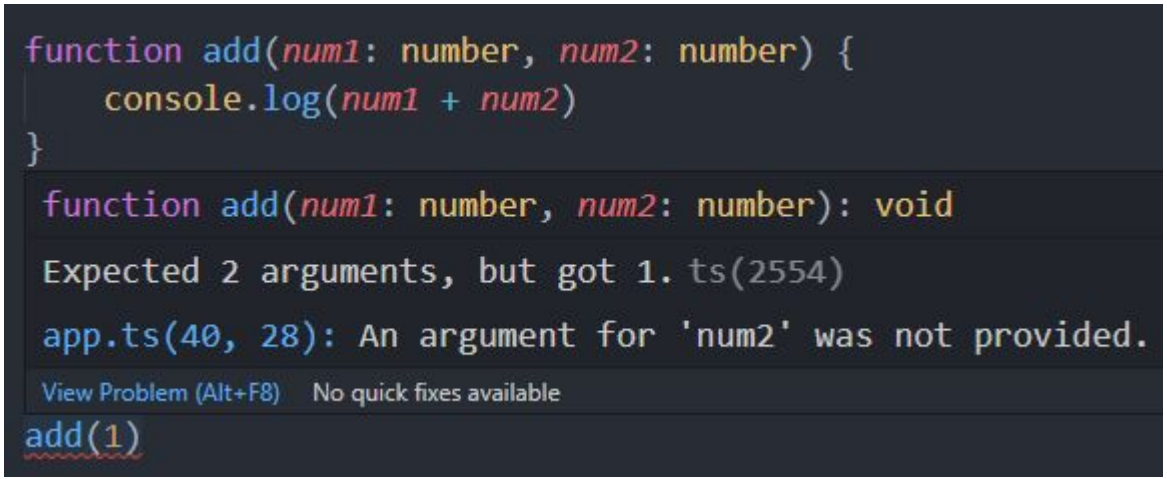
```
interface Person {
    name: string;
    age: number;
    id?: number;
}
```

```
const person: Person = {
    name: "J",
    age: 21,
}

console.log(person.shame)

console.log(person.id.toStr());
```

```
any
Property 'shame' does not exist on type 'Person'.
View Problem (Alt+F8)    Quick Fix... (Ctrl+.)
```

TypeScript Usage - Functions & Documentation

- Typescript catches any invalid function calls
- Provides clear documentation on function usage



TypeScript Usage - Clearer Code

```
function addNumAndSort(list,
    list. any num)
    list.sort()
}
```

- Typescript allows for clearer code without extra documentation
- Documentation with Typescript is more explicit with intended usage

```
function addNumAndSort(list: Array<number>, num: number) {
    list.push(num)
    list.sort()
}
```

```
(method) Array<number>.sort(compareFn?: (a: number, b: number) => number): number[]
```

Sorts an array in place. This method mutates the array and returns a reference to the same array.

@param `compareFn`
Function used to determine the order of the elements. It is expected to return a negative value if first argument is less than second argument, zero if they're equal and a positive value otherwise. If omitted, the elements are sorted in ascending, ASCII character order.

```
[11,2,22,1].sort((a, b) => a - b)
```

Typescript Usage - Not-So Implicit Coercion

- Javascript has many unintended behavior
- Typescript gives compile-time errors to prevent many unintended issues

Javascript

```
"0" == null;                // false
"0" == undefined;           // false
"0" == false;               // true -- UH OH!
"0" == NaN;                 // false
"0" == 0;                   // true
"0" == "";                  // false

false == null;              // false
false == undefined;         // false
false == NaN;               // false
false == 0;                 // true -- UH OH!
false == "";                // true -- UH OH!
false == [];                // true -- UH OH!
false == {};                // false

"" == null;                 // false
"" == undefined;            // false
"" == NaN;                  // false
"" == 0;                    // true -- UH OH!
"" == [];                   // true -- UH OH!
"" == {};                   // false

0 == null;                  // false
0 == undefined;             // false
0 == NaN;                   // false
0 == [];                    // true -- UH OH!
0 == {};                    // false
```

Typescript

```
"0" == null;                // false
"0" == undefined;           // false
"0" == false;               // compile-time error
"0" == NaN;                 // compile-time error
"0" == 0;                   // compile-time error
"0" == "";                  // compile-time error

false == null;              // false
false == undefined;         // false
false == NaN;               // compile-time error
false == 0;                 // compile-time error
false == "";                // compile-time error
false == [];                // compile-time error
false == {};                // compile-time error

"" == null;                 // false
"" == undefined;            // false
"" == NaN;                  // compile-time error
"" == 0;                    // compile-time error
"" == [];                   // compile-time error
"" == {};                   // compile-time error

0 == null;                  // false
0 == undefined;             // false
0 == NaN;                   // compile-time error
0 == [];                    // compile-time error
0 == {};                    // compile-time error
```

Installing Typescript

- Install with
  - `"npm install typescript --save-dev"`
  - `"npm install -g typescript"`
  - `"npx create-react-app my-app --template typescript"`
- Add tsconfig file
  - `"npx tsc --init"`
  - adjust tsconfig.json file to set Typescript behavior
- Can incrementally add Typescript files to existing project

Downsides of Typescript

- Longer to initially set up a project
- Have to write more code
- External Libraries may not have Typescript support
- Compiling takes time

Thank you!

Questions? Comments? Concerns?