[Abdelmoneim Hany Elsayed] [Link]
Team

Features of Typescript beyond static typing.

1- Structural Typing

Structural Equivalence is used to determine compatibility and equivalence between different types. If two types were made using different names but had the same type of features regardless of their name, then the two types are equivalent.

```
1   type Person = {
2     name: string;
3     DOB: Date;
4   };
5
6   type Employee = {
7     name: string;
8     DOB: Date;
9   };
10
11  const person: Person = {
12    name: 'Buzz Lightyear',
13    DOB: new Date(1953, 5, 13)
14  };
15
16  const employee: Employee = person;
17
```

Here in this snippet if in type Employee there were two features with different names but still a string and a Date then line 16 is still valid since there are two features that are a string and Date in both types Person and Employee.

[Nagy Nabil] [LINKEDIN]

"If it walks like a duck and it quacks like a duck, then it must be a duck"

2- add new features to JS like enum and generics

Enum is a programming concept can be found in languages like c++

"an enumerated type is a data type consisting of a set of named values called elements, members, enumeral, or enumerators of the type. The enumerator names are usually identifiers that behave as constants in the language"

Enum code in TS where creating enum is almost like other programing languages

```
132   enum DiallingCodes {
133     EGYPT = 20,
134     USA = 1,
135     ITALY = 39
136   }
137   console.log(`EGYPT dialing code +${DiallingCodes.EGYPT}`);
```

Transpiled JS

```
var DiallingCodes;
(function (DiallingCodes) {
    DiallingCodes[DiallingCodes["EGYPT"] = 20] = "EGYPT";
    DiallingCodes[DiallingCodes["USA"] = 1] = "USA";
    DiallingCodes[DiallingCodes["ITALY"] = 39] = "ITALY";
})(DiallingCodes || (DiallingCodes = {}));
console.log(`EGYPT dialing code +${DiallingCodes.EGYPT}`);
```

[Ahmed Hisham] [https://www.linkedin.com/in/ahmed-hisham-8945441b1/]
Individually

What are the features of Typescript beyond static typing?

General-Features:

Object-Oriented language: TypeScript provides a complete feature of an object-oriented programming language such as classes, interfaces, inheritance, modules, etc. In TypeScript, we can write code for both client-side as well as server-side development.

TypeScript supports JavaScript libraries: TypeScript supports each JavaScript elements. It allows the developers to use existing JavaScript code with the TypeScript. Here, we can use all of the JavaScript frameworks, tools, and other libraries easily.

TypeScript is portable: TypeScript is portable because it can be executed on any browsers, devices, or any operating systems. It can be run in any environment where JavaScript runs on. It is not specific to any virtual-machine for execution.

DOM Manipulation: TypeScript can be used to manipulate the DOM for adding or removing elements similar to JavaScript.

More of a minor-features:

String valued enums:

This is a built-in feature in typescript. It has been long in the making. In fact some of the issues date back to 2014 and earlier. Now in TypeScript version `2.4` we are finally getting them: string valued enums.

Using them is super straight forward:

```
enum Color {
  Red = '#ff0000',
  Green = '#00ff00',
  Blue = '#0000ff'
}
const myFavoriteColor = Color.Green;
let chosenColor = Color.Red;
```

In this case `myFavoriteColor` will be of the value `'#00ff00'` during runtime and it has the type of `Color.Green` since it's a constant. For the `chosenColor`, TypeScript will automatically walk one

step up and assign it the type `Color` since it could change over time. The value assignned during runtime will be `'#ff0000'` as expected.

Transpiling `async/await` to ES5/ES3

Simply add the respective libs to the `lib` property in the `tsconfig.json` and adjust the `target` to your preferred target:

`tsconfig.json`

```
{
  "compilerOptions": {
        "module": "commonjs",
        "target": "es3",
        "lib": ["dom", "es2015", "es2015.promise"]
  }
}
```

From that moment on you can use `async`/`await` simply in your code:

```
async function hello(): Promise<string> {
  const x = await Promise.resolve('foo');
  return x;
}
hello().then((x: string) => {
  console.log(x);
});
```

Reference:

-https://www.javatpoint.com/typescript-features.

-https://moin.world/2017/06/18/10-typescript-features-you-might-not-know