https://github.com/FAQGURU/FAQGURU/blob/master/topics/en/typeScript.md

Mi az a TypeScript, és miért van rá szükségünk?

A JavaScript az egyetlen kliensoldali nyelv, amelyet minden böngésző univerzálisan támogat. De a JavaScript nem a legjobban tervezett nyelv. Ez nem objektumorientált nyelv, nem támogatja az osztályalapú öröklést, a megbízhatatlan dinamikus típus és hiányzik a fordítási idő hibajavítása. A TypeScript pedig megoldja ezeket a problémákat. Más szavakkal, a TypeScript kísérlet a JavaScript problémák „kijavítására".

A TypeScript egy ingyenes és nyílt forráskódú programozási nyelv, amelyet a Microsoft fejlesztett és tart fenn. Ez egy strict JavaScript-készlet, amely opcionális statikus típus és osztályalapú objektum-orientált programozást ad a nyelvhez. A TypeScript meglehetősen könnyen megtanulható és használható a fejlesztők számára, akik ismerik a C #, a Java és az összes erősen típus nyelvet. „A TypeScript egy olyan nyelv, amely egyszerű JavaScript fájlokat generál."

Amint azt a Typescript hivatalos weboldala is elmondta: „A TypeScript lehetővé teszi a JavaScript írását a kívánt módon. A TypeScript egy tipikus JavaScript-készlet, amely egyszerű JavaScript-re áll össze. Bármely böngésző. Bármely szerver. Bármely operációs rendszer. Nyílt forráskód." Ahol a „tipizált" azt jelenti, hogy figyelembe veszi a változók típusait, paramétereit és függvényeit.

Magyarázza el a generikusokat a TypeScript-ben

A generikusok képesek létrehozni egy komponenst vagy függvényt, hogy több típuson is működjenek, nem pedig egyetlen típuson.

```
/** A class definition with a generic parameter */
class Queue<T> {
  private data = [];
  push = (item: T) => this.data.push(item);
  pop = (): T => this.data.shift();
}

const queue = new Queue<number>();
queue.push(0);
queue.push("1"); // ERROR : cannot push a string. Only numbers allowed
```

Mi az a TypeScript, és miért használnám a JavaScript helyett?

A TypeScript a JavaScript egy olyan halmaza, amely elsősorban opcionális statikus típust, osztályokat és interfészeket biztosít. Az egyik nagy előny, hogy lehetővé teszi az IDE-k számára, hogy gazdagabb környezetet biztosítsanak a gyakori hibák észleléséhez a kód beírásakor. Egy nagy JavaScript projekt esetében a TypeScript elfogadása robusztusabb szoftvert eredményezhet, miközben továbbra is telepíthető, ahol egy szokásos JavaScript alkalmazás fut.

Részletesen:

- A TypeScript támogatja az új ECMAScript szabványokat, és összeállítja azokat az Ön által kiválasztott (régebbi) ECMAScript célokhoz. Ez azt jelenti, hogy ma már használhatja az ES2015 és azon kívüli szolgáltatásait, például modulokat, lambda függvényeket, osztályokat, a spread operátort, a destruktúrát.
- A JavaScript kód érvényes TypeScript kód; A TypeScript a JavaScript szuperhalmaza.
- A TypeScript típustámogatást ad a JavaScripthez. A TypeScript típusrendszere viszonylag gazdag, és magában foglalja: interfészeket, enumokat, hibrid típusokat, generikusokat, unió és intersection típusokat, hozzáférés módosítókat és még sok minden mást. A TypeScript egy kicsit megkönnyíti a kódolást és sokkal kevésbé egyértelművé teszi a típuskövetkeztetés használatát.
- A TypeScript fejlesztési tapasztalata nagy előrelépés a JavaScript-hez képest. Az IDE-t valós időben tájékoztatja a TypeScript fordító a gazdag típusú információkról.
- A szigorú nullellenőrzések engedélyezésével (--strictNullChecks compiler flag) a TypeScript fordító nem engedélyezi a undefined hozzárendelését egy változóhoz, hacsak nem kifejezetten deklarálja, hogy nullable típusú.
- A TypeScript használatához build folyamatra van szükség a JavaScript kódra történő fordításhoz. A TypeScript fordító beágyazhatja a sourcemap információkat a létrehozott .js fájlokba, vagy különálló .map fájlokat hozhat létre. Ez lehetővé teszi a brakepoints beállítását és a változók ellenőrzését futás közben közvetlenül a TypeScript kódon.
- A TypeScript nyílt forráskódú (Apache 2 licencelt, lásd: github), és a Microsoft támogatja. Anders Hejlsberg, a C # vezető építésze vezeti a projektet.

Mi a getters / setters a TypeScript-ben?

A TypeScript támogatja a getters / setters, mint az objektum egyik tagjának hozzáférését. Ez lehetővé teszi, hogy finomabb részletességgel ellenőrizhesse, hogy az egyes objektumokhoz miként férhet hozzá a tag.

```
class foo {
  private _bar:boolean = false;

  get bar():boolean {
    return this._bar;
  }
  set bar(theBar:boolean) {
    this._bar = theBar;
  }
}

var myBar = myFoo.bar;  // correct (get)
myFoo.bar = true;  // correct (set)
```

Támogatja a TypeScript minden objektumorientált elvet?

A válasz igen. Az objektumorientált programozásnak 4 fő alapelve van:

- Egységbezárás,
- Öröklés,
- Absztrakció, és

- Polimorfizmus.

Hogyan lehet osztálykonstansokat implementálni a TypeScript-be?

A TypeScriptben az const kulcsszó nem használható osztálytulajdonságok deklarálására. Ezzel a fordító hibát okoz a következővel: "Egy osztálytag nem rendelkezhet a 'const' kulcsszóval."

```
class MyClass {
    readonly myReadonlyProperty = 1;

    myMethod() {
        console.log(this.myReadonlyProperty);
    }
}

new MyClass().myReadonlyProperty = 5; // error, readonly
```

Mi az a TypeScript Map fájl?

A .map fájlok olyan source map fájlok, amelyek segítségével az eszközök leképezhetik a kibocsátott JavaScript kódot és az azt létrehozó TypeScript forrásfájlokat. Sok debugger (pl. Visual Studio vagy a Chrome fejlesztői eszközei) használhatja ezeket a fájlokat, így a TypeScript fájlt hibakeresheti a JavaScript fájl helyett.

Melyek a TypeScript különböző elemei?

A TypeScript főleg 3 összetevője van.

- Nyelv - A fejlesztők számára a legfontosabb az új nyelv. A nyelv új szintaxisból, kulcsszavakból áll, és lehetővé teszi a TypeScript írását.
- Fordító - A TypeScript fordító nyílt forráskódú, platformokon átnyúló és nyílt specifikáció, és TypeScript-be van írva. A Compiler a TypeScript-t JavaScript-be fordítja. És hibát is kibocsát, ha van ilyen. Ez segíthet a különböző fájlok egyetlen kimeneti fájlba konvertálásában és a source map létrehozásában is.
- Nyelvi szolgáltatás - TypeScript nyelvi szolgáltatás, amely az interaktív TypeScript élményt biztosítja a Visual Studio, a VS Code, a Sublime, a TypeScript playground és más szerkesztőkben.

Mi a tipizálás a typescriptben?

A TypeScript főként típusok hozzáadásáról szól a JavaScript-en. Ha olyan külső könyvtárakat használ, mint a jQuery vagy a moment.js, akkor a kódban nincsenek információk a típusokról. Tehát a TypeScript használatához fájlokat is kell kapnia, amelyek leírják a kód típusait. Ezek a típusdeklarációs fájlok, leggyakrabban a .d.ts fájlkiterjesztés nevével. Szerencsére az emberek ilyen típusú deklarációs fájlokat írtak a leggyakoribb javascript könyvtárakhoz.

A Typings csak egy eszköz volt a fájlok telepítésére. A legjobb gyakorlat, hogy csak az npm-et használja.

Hogyan írható opcionálisan statikusan a TypeScript nyelv?

A TypeScript opcionálisan statikusan típusos, vagyis azt kérheti a fordítótól, hogy hagyja figyelmen kívül a változó típusát. Bármely adattípus felhasználásával bármilyen típusú értéket rendelhetünk a változóhoz. A TypeScript nem ad hibát a fordítás során.

```
var unknownType: any = 4;
unknownType = "Okay, I am a string";
unknownType = false; // A boolean.
```

Magyarázza el, hogy mikor kell használni a "declare" kulcsszót a TypeScript-ben

A TypeScript declare kulcsszóval olyan változókat deklarálhatunk, amelyek esetleg nem TypeScript fájlból származnak.

Képzeljük el például, hogy van egy myLibrary nevű könyvtárunk, amelyben nincs TypeScript deklarációs fájl, és a globális névtérben van egy myLibrary nevű névtér. Ha azt a könyvtárat szeretné használni a TypeScript kódban, akkor a következő kódot használhatja:

```
declare var myLibrary;
```

Az a típus, amelyet a TypeScript futásideje ad a myLibrary változónak, any típusú. A probléma itt az, hogy nem lesz Intellisense az adott változóhoz tervezési időben, de a könyvtárat használhatja a kódban. Egy másik lehetőség ugyanarra a viselkedésre a declare kulcsszó használata nélkül, ha csak egy változót használunk any típusúhoz:

```
var myLibrary: any;
```

Mindkét kódpélda ugyanazt a JavaScript kimenetet eredményezi, de a deklarálási példa olvashatóbb és egy ambient deklarációt fejez ki.

Mi az Ambiens a TypeScriptsben, és mikor kell használni?

Az Ambientek vagy az Ambient deklarációk segítségével meg lehet mondani a TypeScript fordítónak, hogy a tényleges forráskód máshol létezik. Az ambient deklarációk segítenek más js könyvtárak zökkenőmentes integrálásában a TypeScript-be.

Az ambient deklarációkat egyezmény szerint egy d.ts kiterjesztésű típusú deklarációs fájlban tárolják. A környezeti változók vagy modulok deklarálásának szintaxisa a következő lesz:

```
declare module Module_Name {
}
```

Az ambient fájlokra hivatkozni kell a kliens TypeScript fájljában, az ábra szerint:

```
/// <reference path = " Sample.d.ts" />
```