

Laborationsrapport

Laboration / Grunderna i TypeScript
Kurskod: DT208G, Programmering i TypeScript

Författare: Mohamed Abokashef, moab2302@student.miun.se
Termin, år: 02, 2024

1.Sammanfattning

jag kommer att i den här labb att göra en analys om Typescript programmering språk, beskriva utvecklings-miljö för det hur det fungerar och hur man kör TypeScript-kod, transpilera TypeScript till "vanlig" JavaScript. Hur man deklarerar variabler med en specifik typ och visar olika typer av variabler som finns samt definierar funktion med typer för parametrar och användning av vanlig och Arrow funktion. jag kommer att beskriva och förklara vad interface och generics är.

2. Innehållsförteckning

1	Sammanfattning.....
2	Innehållsförteckning
3	Typescript Analysen
4	deklarerar variabler
5	definierar funktion.....
6	Beskriva interface.....
7	beskriva generics.....
8	slusats.....
9	källförteckning.....

Typescript Analysen

TypeScript är ett språk för programmering som skapats av Microsoft och publicerades första gången den 2012. Det är ett öppet källkodsprojekt och alla kan använda och utveckla det. TypeScript skapades för att kompensera för bristerna i ren JavaScript när det kommer till att hantera omfattande och svårta kodbasen [1].

Användningsområde: TypeScript är ett språk med statisk typning som omvandlas till vanlig JavaScript. Det inkluderar stöd för typer, klasser och moduler som inte finns i JavaScript. Denna typ av funktion är speciellt användbar för programmerare som vill skapa eller utveckla stora projekten där det är viktigt med kodens underhåll och skalbarhet.

Vad gör det användbart?

1-Vad det gör användbart för TypeScript erbjuder statisk typning, det betyder att felaktiga typer kan upptäckas under kompilering i stället för vid körning. Det resulterar i minskad förekomst av buggar och ökad stabilitet i koden.

2-Många olika programmeringsmiljöer IDE:er såsom Visual Studio Code, och Atom erbjuder utmärkt support för TypeScript. Det innefattar funktioner som automatisk komplettering, snabb navigering och omstrukturering.

3-Genom att inkludera olika typer och klasser blir koden mer begriplig och enklare att hantera. Det blir också tydligare hur olika delar av koden kommer att användas och interagera med varandra.

4-TypeScript har inbyggt stöd för koncept som gränssnitt och moduler, vilket gör koden mer strukturerad och enklare att använda igen.

Exempel på användbarhet:

```
1 function add(n1, n2) {  
2   return n1 + n2;  
3 }  
4  
5 console.log(add(10, 20));  
6
```

Figur 1

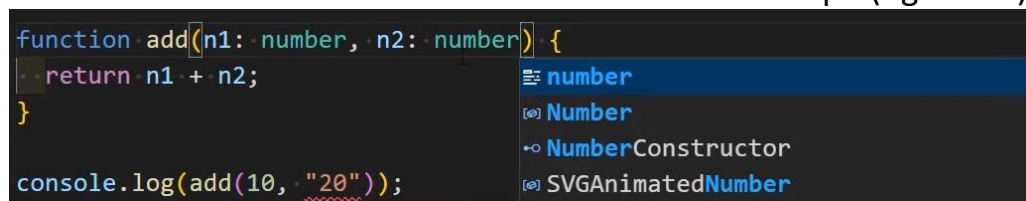
Det är en js funktion som tar två numer och addera dem, resultat kommer att bli 30 i så fall (figure 1) men om vi gör 20 sträng i stället av nummer vad kommer att hända i så fall, js kod kommer att sammanfoga dem och resultat kommer att bli 1020 och type av variabel kommer att bli sträng i så fall (figure 2).

```
1 function add(n1, n2) {  
2   return n1 + n2;  
3 }  
4  
5 console.log(add(10, "20"));
```

Figur 2

Men det kommer inte att hända i typescript eftersom kommer att ge error innan man kör koden som i den exempel(figure 3):

```
function add(n1: number, n2: number) {  
  return n1 + n2;  
}  
  
console.log(add(10, "20"));
```



Figur 3

Hur deklarerar du en variabel med en specifik typ i TypeScript? Vilka olika typer finns det?

Vi kan deklarera i TypeScript en variabel med en specifik typ genom att använda kolonnotationen efter variabelnamnet [2]. några exempel på det: numeriska värden `let antal: number = 5;`, strängtext `let variabel: string = "typescript kurs";`, boolean som ge true eller false värde `let truth: boolean = true;`, array i den exempel kommer att

skapa en array av sträng värde 'let data: string[] = ["one", "two", "three"];' Enum definierar och sätter upp namngivna konstanter och det finns numeriska, sträng och heterogeneous[string+ number] Enums, exempel: enum kurs {

Typescript,

Javascript,

webbutveckling

} let nuvarandeKurs: kurs = kurs.Typescript; , i typescript kan vi deklarera även med any vilket tillåter definiera vilka värde som helst.

“let värde: any = ”här väljer vi vilka värde vi vill”.

3.Hur definierar du en funktion med specifika typer för parametrar och returvärden i TypeScript?

Ge kod-exempel, och använd både "vanlig" syntax samt "arrow function"-syntax.

Vi kan definiera en funktion i typescript med vilka typer för parametrar och returvärden genom att specificera typerna efter parameterlistan och returvärden efter kolon [3] och här ett exempel på det [figure 4]:

Vanlig function:

```
est.ts > ...  
Click here to ask Blackbox to help you code faster  
  
let validValue: number = 10;  
function showData(name: string, myAge: number, city: string) {  
  if (validValue = 10) {  
    return `mitt namn ${ name }, jag är ${ myAge } gammal, bor i ${ city }`;  
  }  
}  
console.log(showData('mohamed', 30, 'Helsingborg'));  
  
// Arrow function:  
  
const showData = (name: string, myAge: number, city: string) => {  
  if (validValue = 10) {  
    return `mitt namn ${name}, jag är ${myAge} gammal, bor i ${city}`;  
  }  
};  
console.log(showData("mohamed", 30, "Helsingborg"));
```

Figur 4

4. Vad är en "interface" i TypeScript och hur används det?

Ge kod-exempel.

interface definierar som en abstrakt beskrivning av en samling relaterade attribut och funktioner som en typ kan innehålla.

Det används för att komma överens om hur olika delar av koden ska samverka med varandra. Gränssnittet möjliggör skapandet av typer som sedan kan användas för att säkert specificera strukturer av objekt. vilket gör koden att det kan återanvända [4]. Ett exempel: se figure 5



```
Click here to ask Blackbox to help you code faster

interface me {
  name: string,
  age: number,
  country: string
}

let me: me = {
  name: "Mohamed",
  age: 30,
  country: "sweden"
}

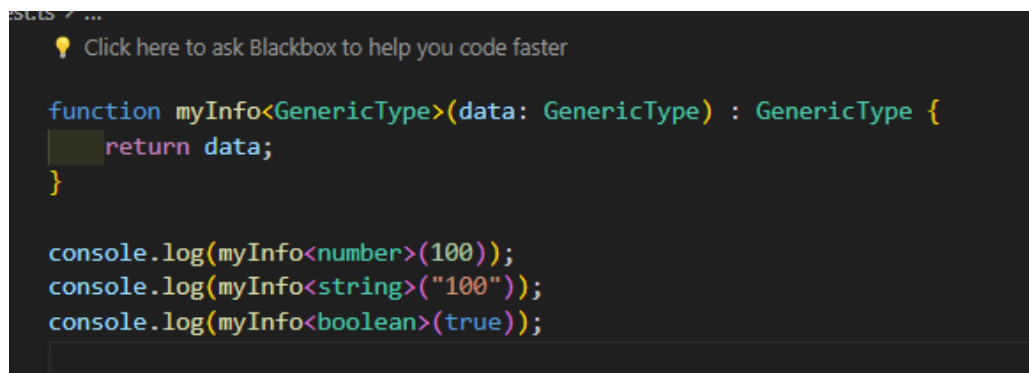
function myInfo(data: me) {
  console.log(`my name is ${data.name} and my age is ${data.age}, live in ${data.country}`);
}
```

Figur 5

5. Vad är "generics" i TypeScript och varför är de användbara?

Ge kod-exempel.

Generics är ett verktyg som hjälper oss att skriva en kod som kan återanvända den, det tillåter oss behandla multipler typer utan att använda any type[4]. Se figure 6 i exempel kan vi styra typer utan att använda any-type eller att ändra något i funktion själv.



```
Click here to ask Blackbox to help you code faster

function myInfo<GenericType>(data: GenericType) : GenericType {
  return data;
}

console.log(myInfo<number>(100));
console.log(myInfo<string>("100"));
console.log(myInfo<boolean>(true));
```

Figur 6

Resultat

TypeScript är ett effektivt programmeringsspråk för att skriva kod på ett säkrare och mer organiserat sätt, vilket är särskilt fördelaktigt i stora projekten.

Källförteckning

1. Wikipedia contributors. (2024, March 10). *TypeScript*. Wikipedia, The Free Encyclopedia. <https://en.wikipedia.org/w/index.php?title=TypeScript&oldid=1213047707>
2. *Everyday types*. (n.d.). Typescriptlang.org. Retrieved March 20, 2024, from <https://www.typescriptlang.org/docs/handbook/2/everyday-types.html>
3. *More on Functions*. (n.d.). Typescriptlang.org. Retrieved March 20, 2024, from <https://www.typescriptlang.org/docs/handbook/2/functions.html>
4. *Generics*. (n.d.). Typescriptlang.org. Retrieved March 20, 2024, from <https://www.typescriptlang.org/docs/handbook/2/generics.html>