Información general de tipos en TypeScript

5 minutos

La principal ventaja de TypeScript es que permite agregar tipos estáticos al código de JavaScript. Los tipos colocan restricciones estáticas en entidades de programa, como funciones, variables y propiedades, de tal modo que los compiladores y las herramientas de desarrollo puedan ofrecer una mejor comprobación y asistencia durante el desarrollo.

El sistema de tipos en tiempo de compilación estático de TypeScript modela estrechamente el sistema de tipos en tiempo de ejecución dinámico de JavaScript. El sistema de tipos en tiempo de compilación estático le permite expresar de forma precisa las relaciones de tipos que se prevé que existan cuando se ejecutan sus programas. A continuación, haga que el compilador de TypeScript valide esas suposiciones. El análisis de tipos de TypeScript se produce por completo en tiempo de compilación y no agrega sobrecarga en tiempo de ejecución a la ejecución del programa.

Los tipos estáticos también proporcionan una manera de documentar mejor la intención del código, lo que le permitirá a usted y a otros desarrolladores comprenderlo.

Declaración de variables let y const

ECMAScript 2015 ha agregado las palabra clave `let` y `const` para la declaración de variables en JavaScript, que ha eliminado algunos de los problemas asociados con la palabra clave `var` en versiones anteriores. Este cambio permite declarar variables con ámbito de nivel de bloque y evita que se declare la misma variable varias veces.

TypeScript fomenta el uso de las palabras clave `let` y `const` para las declaraciones de variables.

> **Nota**
>
> Como recordatorio, la diferencia entre ellas es que las declaraciones `let` se pueden realizar sin inicialización, mientras que las declaraciones `const` siempre se inicializan con un valor. Y las declaraciones `const`, una vez asignadas, nunca se pueden volver a asignar.

Ejercicio: Inferencia de tipo en TypeScript

Puede asociar tipos con variables mediante anotaciones de tipo explícitas o la inferencia de tipos implícita.

Aunque se recomienda, las anotaciones de tipo explícitas son opcionales en TypeScript. Para declarar un tipo explícito, use la sintaxis `variableName: type`. Esta instrucción `let myVariable: number` declara la variable como un tipo de número sin inicializarla. Como alternativa, también puede inicializar la variable mediante `let myVariable: number = 10`.

Para implicar el tipo de variable mediante la inferencia de tipos, use el mismo formato que se usa en JavaScript. Por ejemplo, `let myVariable = 10` infiere que la variable es de tipo `number` porque se inicializa con el valor `10`.

Vamos a abrir el área de juegos para ver cómo funciona esto.

1. Escriba las declaraciones de variable siguientes:

   TypeScript

   ```
   let x: number;   //* Explicitly declares x as a number type
   let y = 1;       //* Implicitly declares y as a number type
   let z;           //* Declares z without initializing it
   ```

2. TypeScript ahora trata la variable `x` como un tipo `number`. TypeScript también infiere el tipo de `y` para que sea un tipo `number` porque es el tipo del valor que se usa para inicializarlo. Pero, ¿qué ocurre si intenta asignar un tipo de valor diferente? ¿Y qué ocurre con la variable z?

3. Abra la pestaña **Errores** en el Área de juegos para que pueda supervisar los errores.

4. Escriba `x = 1`. Esta declaración debería funcionar según lo esperado, sin errores.

5. Escriba `x = "one"`. Tal como se esperaba, esta declaración genera el error **El tipo "string" no se puede asignar al tipo "number"** porque la comprobación de tipos estáticos no permite que un elemento `string` se asigne a la variable.

6. Escriba `y = "one"`. Verá que se produce el mismo error porque TypeScript ha inferido que "y" es de tipo `number`.

7. Escriba el nombre de la variable `y` seguido de un punto y observará que hay algo más. Aunque no se ha especificado de forma explícita un tipo para la variable `y`, IntelliSense proporciona métodos que solo se aplican a un tipo `number`.

8. Escriba `z = 1` y `z = "one"`. TypeScript ha aceptado ambos valores, pero ¿por qué? Estas declaraciones funcionan de la misma manera que en JavaScript, ya que la variable `z` ahora puede aceptar cualquier valor asignado. TypeScript ha inferido que `z` es de tipo `any` porque no ha asignado un tipo o lo ha inicializado cuando se declaró. Más adelante, aprenderá más sobre el tipo `any`.

Aunque puede inferir de forma implícita los tipos mediante la inferencia de tipos en TypeScript, ¿debería? Mediante la inferencia de tipos, se obtienen algunas de las ventajas de la comprobación de tipos estáticos y de IntelliSense, y permite migrar gradualmente a declaraciones de tipos explícitas en los proyectos. Sin embargo, las declaraciones de tipos explícitas también proporcionan una manera de documentar mejor la intención del código y proporcionar una ruta de acceso más deliberada.

Tipos y subtipos en TypeScript

Antes de profundizar en el uso de tipos para la declaración de variables, echemos un vistazo a los tipos y subtipos en TypeScript.

Cualquier tipo

Todos los tipos en TypeScript son subtipos de un único tipo principal denominado tipo `any`. `any` es un tipo que puede representar cualquier valor de JavaScript sin restricciones. Todos los demás tipos se clasifican como tipos primitivos, tipos de objeto o parámetros de tipo. Estos tipos presentan diversas restricciones estáticas en sus valores.

Tipos primitivos

Los tipos primitivos son los tipos `boolean`, `number`, `string`, `void`, `null` y `undefined`, junto con enumeración definida por el usuario o tipos `enum`. El tipo `void` existe únicamente para indicar la ausencia de un valor, como en una función sin ningún valor devuelto. Los tipos `null` y `undefined` son subtipos de todos los demás tipos. No es posible hacer referencia explícita a los tipos `null` y `undefined`. Solo se puede hacer referencia a los valores de esos tipos mediante los literales `null` y `undefined`.

Tipos de objeto y parámetros de tipo

Los tipos de objeto son todos los tipos de clase, interfaz, matriz y literal (todo lo que no sea un tipo primitivo).

Los tipos de clase e interfaz se introducen mediante las declaraciones de clase e interfaz, y se hace referencia a ellos con el nombre que se les ha asignado en sus declaraciones. Los tipos de clase e interfaz pueden ser tipos genéricos que tienen uno o más parámetros de tipo. Obtendrá más información sobre estos tipos de objeto en módulos posteriores.

Siguiente unidad: Tipos primitivos en TypeScript

Continuar >

¿Qué tal lo estamos haciendo? ☆ ☆ ☆ ☆ ☆