类型推论

介绍

这节介绍TypeScript里的类型推论。即，类型是在哪里如何被推断的。

基础

TypeScript里，在有些没有明确指出类型的地方，类型推论会帮助提供类型。如下面的例子

```
let x = 3;
```

变量 x 的类型被推断为数字。 这种推断发生在初始化变量和成员，设置默认参数值和决定函数返回值时。

大多数情况下，类型推论是直截了当地。 后面的小节，我们会浏览类型推论时的细微差别。

最佳通用类型

当需要从几个表达式中推断类型时候，会使用这些表达式的类型来推断出一个最合适的通用类型。例如，

```
let x = [0, 1, null];
```

为了推断 x 的类型，我们必须考虑所有元素的类型。 这里有两种选择： `number` 和 `null` 。 计算通用类型算法会考虑所有的候选类型，并给出一个兼容所有候选类型的类型。

由于最终的通用类型取自候选类型，有些时候候选类型共享相同的通用类型，但是却没有一个类型能做为所有候选类型的类型。例如：

```
let zoo = [new Rhino(), new Elephant(), new Snake()];
```

这里，我们想让zoo被推断为 `Animal[]` 类型，但是这个数组里没有对象是 `Animal` 类型的，因此不能推断出这个结果。 为了更正，当候选类型不能使用的时候我们需要明确的指出类型：

```
let zoo: Animal[] = [new Rhino(), new Elephant(), new Snake()];
```

如果没有找到最佳通用类型的话，类型推论的结果是空对象类型， `{}` 。 因为这个类型没有任何成员，所以访问其成员的时候会报错。

上下文类型

TypeScript类型推论也可能按照相反的方向进行。 这被叫做“按上下文归类”。按上下文归类会发生在表达式的类型与所处的位置相关时。比如：

```
window.onmousedown = function(mouseEvent) {
    console.log(mouseEvent.buton);  //<- Error
};
```

这个例子会得到一个类型错误，TypeScript类型检查器使用 `Window.onmousedown` 函数的类型来推断右边函数表达式的类型。 因此，就能推断出 `mouseEvent` 参数的类型了。 如果函数表达式不是在上下文类型的位置， `mouseEvent` 参数的类型需要指定为 `any` ，这样也不会报错了。

如果上下文类型表达式包含了明确的类型信息，上下文的类型被忽略。 重写上面的例子：

```
window.onmousedown = function(mouseEvent: any) {
    console.log(mouseEvent.buton);  //<- Now, no error is given
};
```

这个函数表达式有明确的参数类型注解，上下文类型被忽略。 这样的话就不报错了，因为这里不会使用到上下文类型。

上下文归类会在很多情况下使用到。 通常包含函数的参数，赋值表达式的右边，类型断言，对象成员和数组字面量和返回值语句。 上下文类型也会做为最佳通用类型的候选类型。比如：

```
function createZoo(): Animal[] {
    return [new Rhino(), new Elephant(), new Snake()];
}
```

这个例子里，最佳通用类型有4个候选者： `Animal` ， `Rhino` ， `Elephant` 和 `Snake` 。 当然， `Animal` 会被做为最佳通用类型。