Generic Default

This lesson explains how to assign a default value to generic type.

TypeScript allows defining default type for generic, also known as “generic parameter defaults.” The syntax is intuitive which is the equal sign of the generic type. The generic type parameter doesn’t need to be explicit when an optional value is provided. TypeScript can infer its type.

```
interface MyGenericWithDefault<T = string> {
    myTypeWhichIsStringIfNotSpecified: T;
}
const myGeneric1: MyGenericWithDefault<number> = { myTypeWhichIsStringIfNotSpecified: 1 };
const myGeneric2: MyGenericWithDefault = { myTypeWhichIsStringIfNotSpecified: "string" };
const myGeneric3: MyGenericWithDefault<string> = { myTypeWhichIsStringIfNotSpecified: "string
```

Like the defaults of a function’s parameters, default types can only be used after required types. That means that specifying non-default generic types after a specified default doesn’t compile.

```
interface MyGenericWithDefaults<T = string, Y> {} // Doesn't compile
```

There is a special case with generic and interfaces. Interfaces defined in two different places bring the optional value from one interface into the second one while the schema is being merged.

In the following example, there are three interfaces with three different generic levels. TypeScript then merges the interfaces and its result is the simplest definition, which is defaulted to a `string` that is accessible by all interface definitions – even the ones that don’t have the generic angle brackets.

```
interface InterfaceGenericDefinedTwoPlace<T = string> {
    myProp: T;
}
interface InterfaceGenericDefinedTwoPlace<T> {}
```

```
interface InterfaceGenericDefinedTwoPlace {}
```

To conclude, generic must have a type defined between the `<` and `>` symbols. An empty generic has not been valid since **TypeScript 2.3**.

The following code does not compile if we remove the `T = string` from the first declaration.

```
interface InterfaceGenericDefinedTwoPlace<T = string> {
    myProp: T;
}
interface InterfaceGenericDefinedTwoPlace<T> {}
interface InterfaceGenericDefinedTwoPlace {}

let x: InterfaceGenericDefinedTwoPlace = { myProp: "stringHere" };
```