auto and typeof

This lesson will teach you about the use of auto and typeof in D language.

WE'LL COVER THE FOLLOWING

auto

At times, we have to declare variables that need type declaration on both sides of the `=` operator. For example, when declaring a *file variable* (don't get confused with the type of the variable, we will cover this variable type in a later chapter), we have to repeat the name of the type on both sides of the `=` operator:

```
File file = File("student_records", "w");
```

This looks redundant. Also, it would be cumbersome and error-prone if the type names were long.

```
VeryLongTypeName var = VeryLongTypeName(/* ... */);
```

However, specifying the type on the left-hand side is not necessary if we use the `auto` keyword. In this case, the compiler infers the type of the left-hand side from the expression on the right-hand side. For the compiler to infer the type, the `auto` keyword can be used:

```
auto var = VeryLongTypeName(/* ... */);
```

Few examples of `auto`:

```
auto duration = 42;
```

```
auto distance = 1.2;

auto greeting = "Hello";
auto vehicle = BeautifulBicycle("blue");
```

Although `auto` is the abbreviation of *automatic*, it does not come from the automatic type inference. It represents an **automatic storage class**, which is a concept regarding the lifetime of variables. `auto` is used when no other specifier is appropriate. For example, the following definition does not use `auto`:

```
immutable i = 42;
```

Above, the compiler infers the type of `i` as an immutable `int` above. (We will learn about the `immutable` type in a later chapter.)

`typeof`

`typeof` provides the type of expression (including single variables, objects, literals, etc.) without actually evaluating that expression.

The following is an example of how `typeof` can be used to specify a type without explicitly spelling it out:

```
int value = 100; // already defined as 'int'
typeof(value) value2; // means "type of value"
typeof(100) value3; // means "type of literal 100"
```

The last two variable definitions above are equivalent to the following:

```
int value2;
int value3;
```

It is obvious that `typeof` is not needed in situations when the actual types are known. Instead, you would typically use `typeof` in more elaborate scenarios, where you want the type of your variables to be consistent with some other piece of code whose type can vary. This keyword is especially useful in *templates* and *mixins*.

In the next lesson, you will find a coding challenge related to the use of

In the next lesson, you will find a coding challenge related to the use of variables.