

# TypeScript object Type

```
If this site saves you hours of work, please

whitelist it in your ad blocker (*) to

support us (*)

in creating more helpful and free content

in the future.
```

**Summary**: in this tutorial, you'll learn about the TypeScript object type and how to write more accurate object type declarations.

## Introduction to TypeScript object type

The TypeScript object type represents all values that are not in primitive types.

The following are primitive types in TypeScript:

- number
- bigint
- string
- boolean
- null
- undefined
- symbol

The following shows how to declare a variable that holds an object:

```
let employee: object;

employee = {
    firstName: 'John',
    lastName: 'Doe',
    age: 25,
```

```
jobTitle: 'Web Developer'
};
console.log(employee);
```

Output:

```
firstName: 'John',
  lastName: 'Doe',
  age: 25,
  jobTitle: 'Web Developer'
}
```

If you reassign a primitive value to the employee object, you'll get an error:

```
employee = "Jane";
```

Error:

```
error TS2322: Type '"Jane"' is not assignable to type 'object'.
```

The employee object is an object type with a fixed list of properties. If you attempt to access a property that doesn't exist on the employee object, you'll get an error:

```
console.log(employee.hireDate);
```

Error:

```
error TS2339: Property 'hireDate' does not exist on type 'object'.
```

Note that the above statement works perfectly fine in JavaScript and returns undefined instead.

To explicitly specify properties of the <a href="mailto:employee">employee</a> object, you first use the following syntax to declare the <a href="mailto:employee">employee</a> object:

```
let employee: {
    firstName: string;
    lastName: string;
    age: number;
    jobTitle: string;
};
```

And then assign the employee object to a literal object with the described properties:

```
employee = {
    firstName: 'John',
    lastName: 'Doe',
    age: 25,
    jobTitle: 'Web Developer'
};
```

Or you can combine both syntaxes in the same statement like this:

```
let employee: {
    firstName: string;
    lastName: string;
    age: number;
    jobTitle: string;
} = {
    firstName: 'John',
    lastName: 'Doe',
    age: 25,
    jobTitle: 'Web Developer'
};
```

### object vs. Object

TypeScript has another type called **Object** with the letter **O** in uppercase. It's important to understand the differences between them.

The object type represents all non-primitive values while the Object type describes the functionality of all objects.

For example, the <code>Object</code> type has the <code>toString()</code> and <code>valueOf()</code> methods that can be accessible by any object.

#### The empty type {}

TypeScript has another type called empty type denoted by {} , which is quite similar to the object type.

The empty type {} describes an object that has no property on its own. If you try to access a property on such an object, TypeScript will issue a compile-time error:

```
let vacant: {};
vacant.firstName = 'John';
```

Error:

```
error TS2339: Property 'firstName' does not exist on type '{}'.
```

But you can access all properties and methods declared on the Object type, which is available on the object via the prototype chain:

```
let vacant: {} = {};
console.log(vacant.toString());
```

Output:

```
[object Object]
```

## **Summary**

- The TypeScript object type represents any value that is not a primitive value.
- The Object type, however, describes functionality that is available on all objects.

• The empty type {} refers to an object that has no property on its own.