# TypeScript String Literal Types



**Summary**: in this tutorial, you will learn about the TypeScript string literal types that define a type that accepts a specified string literal.

The string literal types allow you to define a type that accepts only one specified string literal.

The following defines a string literal type that accepts a literal string `'click'`:

```
let click: 'click';
```

The `click` is a string literal type that accepts only the string-literal `'click'`. If you assign the literal string `'click'` to the `click`, it will be valid:

```
click = 'click'; // valid
```

However, when you assign another string literal to the `click`, the TypeScript compiler will issue an error. For example:

```
click = 'dblclick'; // compiler error
```

Error:

```
Type '"dblclick"' is not assignable to type '"click"'.
```

The string literal type is useful to limit a possible string value that a variable can store.

The string literal types can combine nicely with the union types to define a finite set of string literal values for a variable:

```
let mouseEvent: 'click' | 'dblclick' | 'mouseup' | 'mousedown';
mouseEvent = 'click'; // valid
mouseEvent = 'dblclick'; // valid
mouseEvent = 'mouseup'; // valid
mouseEvent = 'mousedown'; // valid
mouseEvent = 'mouseover'; // compiler error
```

If you use the string literal types in multiple places, they will be verbose.

To avoid this, you can use the type aliases. For example:

```
type MyMouseEvent = 'click' | 'dblclick' | 'mouseup' | 'mousedown';
let mouseEvent: MyMouseEvent;
mouseEvent = 'click'; // valid
mouseEvent = 'dblclick'; // valid
mouseEvent = 'mouseup'; // valid
mouseEvent = 'mousedown'; // valid
mouseEvent = 'mouseover'; // compiler error

let anotherEvent: MyMouseEvent;
```

## Summary

- A TypeScript string literal type defines a type that accepts specified string literal.
- Use the string literal types with union types and type aliases to define types that accept a finite set of string literals.