JavaScript Primitive Wrapper Types

Summary: in this tutorial, you will learn about the primitive wrapper types including Boolean, String, and Number.

Introduction to primitive wrapper types

JavaScript provides three primitive wrapper types: Boolean, Number, and String types.

The primitive wrapper types make it easier to use primitive values including booleans, numbers, and strings.

See the following example:

```
let language = 'JavaScript';
let s = language.substring(4);
console.log(s);  // Script
```

In this example, The variable `language` holds a primitive string value. It doesn't have any method like `substring()`. However, the above code works perfectly.

When you call a method on a variable that holds a number, a string, or a boolean, JavaScript performs the following steps behind the scenes:

- Create an object of a corresponding type.
- Call a specific method on the instance.
- Delete the instance immediately.

So the following code

```
let language = 'JavaScript';
let str = language.substring(4);
```

is technically equivalent to the following code:

```
let language = 'JavaScript';
// behind the scenes of the language.substring(4);
let tmp = new String(language);
str = temp.substring(4);
temp = null;
```

Primitive wrapper types vs. reference types

When you create an object of a reference type using the `new` operator, the object will stay in the memory until it goes out of scope.

The following variable `s` will stay on the heap until it goes out of the scope:

```
let s = new String('JavaScript');
console.log(s);
```

However, an automatically created primitive wrapper object exists for one line of code only. See the following example:

```
let s = 'JavaScript';
s.language = 'ECMAScript';
console.log(s.language); // undefined
```

In this example, we attempted to access the `language` property of the `s` variable and received a value of `undefined` instead of `'ECMAScript'`:

```
console.log(s.language); // undefined
```

The reason is that the following code creates a `String` object and assigns a value to the `language` property.

```
s.language = 'ECMAScript';
```

However, the String object with the `language` property only exists during the execution of this line of code.

It’s not recommended to explicitly create primitive wrapper objects like the following:

```
let n = new Number(10);
let s = new String('JS');
let b = new Boolean(false);
```

However, you should know which methods are available for a primitive value in order to manipulate it more effectively.

In this tutorial, you have learned about the JavaScript primitive wrapper types for Booleans, numbers, and strings.