Object Literal Syntax Extensions in ES6

Summary: in this tutorial, you will learn about the syntax extensions of the object literal in ES6 that make your code cleaner and more flexible.

The object literal is one of the most popular patterns for creating objects in JavaScript because of its simplicity. ES6 makes the object literal more succinct and powerful by extending the syntax in some ways.

Object property initializer shorthand

Before ES6, an object literal is a collection of name-value pairs. For example:

```
function createMachine(name, status) {
    return {
        name: name,
        status: status
    };
}
```

The `createMachine()` function takes two arguments `name` and `status` and returns a new object literal with two properties: `name` and `status` .

The `name` and `status` properties take the values of the `name` and `status` parameters. This syntax looks redundant because `name` and `status` mentioned twice in both the `name` and `value` of properties.

ES6 allows you to eliminate duplication when a property of an object is the same as the local variable name by including the name without a colon and value.

For example, you can rewrite the `createMachine()` function in ES6 as follows:

```
function createMachine(name, status) {
    return {
        name,
        status
    };
}
```

Internally, when a property of an object literal only has a name, the JavaScript engine searches for a variable with the same name in the surrounding scope. If the JavaScript engine can find one, it assigns the property the value of the variable.

In this example, the JavaScript engine assigns the `name` and `status` property values of the `name` and `status` arguments.

Similarly, you can construct an object literal from local variables as shown in the following example:

```
let name = 'Computer',
    status = 'On';

let machine = {
   name,
   status
};
```

Computed property name

Before ES6, you could use the square brackets(`[]`) to enable the **computed property names** for the properties on objects.

The square brackets allow you to use the string literals and variables as the property names.

See the following example:

```
let name = 'machine name';
let machine = {
```

```
    [name]: 'server',
    'machine hours': 10000
};

console.log(machine[name]); // server
console.log(machine['machine hours']); // 10000
```

The `name` variable was initialized to a value of `'machine name'`. Since both properties of the `machine` object contain a space, you can only reference them using the square brackets.

In ES6, the computed property name is a part of the object literal syntax, and it uses the square bracket notation.

When a property name is placed inside the square brackets, the JavaScript engine evaluates it as a string. It means that you can use an expression as a property name. For example:

```
let prefix = 'machine';
let machine = {
    [prefix + ' name']: 'server',
    [prefix + ' hours']: 10000
};

console.log(machine['machine name']); // server
console.log(machine['machine hours']); // 10000
```

The `machine` object's properties evaluated to `'machine name'` and `'machine hours'`, therefore you can reference them as the properties of the `machine` object.

Concise method syntax

Before ES6, when defining a method for an object literal, you need to specify the name and full function definition as shown in the following example:

```
let server = {
        name: "Server",
        restart: function () {
                console.log("The" + this.name + " is restarting...");
        }
};
```

ES6 makes the syntax for making a method of the object literal more succinct by removing the colon (:) and the `function` keyword.

The following example rewrites the `server` object above using the ES6 syntax.

```
let server = {
    name: 'Server',
    restart() {
        console.log("The" + this.name + " is restarting...");
    }
};
```

This shorthand syntax is also known as the **concise method syntax**. It's valid to have spaces in the property name. For example:

```
let server = {
    name: 'Server',
    restart() {
        console.log("The " + this.name + " is restarting...");
    },
    'starting up'() {
        console.log("The " +  this.name + " is starting up!");
    }
};

server['starting up']();
```

In this example, the method `'starting up'` has spaces in its name. To call the method, you use the following syntax:

```
object_name['property name']();
```

In this tutorial, you have learned how to use some new object literal syntax extensions in ES6 including property initializer shorthand, computed properties, and concise method syntax.