JavaScript Computed Properties

Summary: in this tutorial, you'll learn about the JavaScript computed properties introduced in ES6.

Introduction to JavaScript Computed Properties

Computed properties allow you to create object properties dynamically using an expression in square brackets `[]`.

Here's the syntax of the computed property:

```
let propertyName = 'dynamicPropertyName';
const obj = {
    [propertyName] : value
}
```

In this syntax:

- `propertyName` is a variable that stores the property name.
- `[propertyName]` is a computed property name of the object literal `obj`. The property name is derived from the value of the `propertyName` variable.
- `value` is the value of the computed property, which can be any valid JavaScript expression.

Once you define a computed property, you can access it as follows:

```
obj.dynamicPropertyName
```

JavaScript Computed Property examples

Let's take some examples of using the JavaScript computed properties.

1) Basic JavaScript computed property example

The following example shows how to use a computed property name:

```
let propName = 'c';

const rank = {
  a: 1,
  b: 2,
  [propName]: 3,
};

console.log(rank.c); // 3
```

In this example, the `[propName]` is a computed property of the `rank` object. The property name is derived from the value of the `propName` variable.

When you access `c` property of the `rank` object, JavaScript evaluates `propName` and returns the property's value.

2) Using a computed property in a class

Like an object literal, you can use computed properties for getters and setters of a class. For example:

```
const name = 'fullName';

class Person {
  constructor(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
  get [name]() {
    return `${this.firstName} ${this.lastName}`;
  }
}
```

```
let person = new Person('John', 'Doe');
console.log(person.fullName);
```

Output:

```
John Doe
```

How it works:

The `get[name]` is a computed property name of a getter of the `Person` class. At runtime, when you access the `fullName` property, the person object calls the getter and returns the full name.

3) Creating an object from a key/value pair

The following example shows how to use computed properties to create an object from a key/value pair:

```
const createObject = (key, value) => {
  return {
    [key]: value,
  };
};

const person = createObject('name', 'John');
console.log(person);
```

Output:

```
{ name: 'John' }
```

Note that without computed property, you have to create an object first, and use bracket notation to assign property to value like this:

```
const createObject = (key, value) => {
  let obj = {};
  obj[key] = value;
```

```
  return obj;
};

const person = createObject('name', 'John');
console.log(person);
```

Output:

```
{ name: 'John' }
```

Summary

- Computed properties allow you to use the values of expressions as property names of an object.