Understanding Own Properties of an Object in JavaScript

Summary: in this tutorial, you will learn about the **own** properties of an object in JavaScript.

In JavaScript, an object is a collection of properties, where each property is a key-value pair.

This example creates a new object called `person` using an object initializer:

```
const person = {
    firstName: 'John',
    lastName: 'Doe'
};
```

The `person` object has two properties: `firstName` and `lastName`.

JavaScript uses prototypal inheritance. Therefore, a property of an object can be either **own** or **inherited**.

A property that is defined directly on an object is **own** while a property that the object receives from its prototype is **inherited**.

The following creates an object called `employee` that inherits from the `person` object:

```
const employee = Object.create(person, {
    job: {
        value: 'JS Developer',
        enumerable: true
    }
});
```

The `employee` object has its own property `job`, and inherits `firstName` and `lastName` properties from its prototype `person`.

The `hasOwnProperty()` method returns `true` if a property is own. For example:

```
console.log(employee.hasOwnProperty('job')); // => true
console.log(employee.hasOwnProperty('firstName')); // => false
console.log(employee.hasOwnProperty('lastName')); // => false
console.log(employee.hasOwnProperty('ssn')); // => false
```

Summary

- A property that is directly defined on an object is an own property.
- The `obj.hasOwnProperty()` method determines whether or not a property is own.