JavaScript Objects

In JavaScript, an object is a collection of properties, defined as a key-value pair. Each property has a key and a value. The property key can be a string and the property value can be any valid value.

To create an object, you use the object literal syntax. For example, the following snippet creates an empty object:

```
let empty = {};
```

To create an object with properties, you use the `key : value` syntax. For example, the following snippet creates a `person` object:

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

The `person` object has two properties `firstName` and `lastName` with the corresponding values `'John'` and `'Doe'`.

Accessing properties

To access a property of an object, you use one of two notations: the dot notation and array-like notation.

1) The dot notation (.)

The following illustrates how to use the dot notation to access a property of an object:

```
objectName.propertyName
```

For example, to access the `firstName` property of the `person` object, you use the following expression:

```
person.firstName
```

The following snippet creates a `person` object and shows the first name and last name on the Console:

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

console.log(person.firstName);
console.log(person.lastName);
```

2) Array-like notation ([ ])

The following illustrates how to access the value of an object's property via the array-like notation:

```
objectName['propertyName'];
```

For example:

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

console.log(person['firstName']);
console.log(person['lastName']);
```

When a property name contains spaces, you need to place it inside quotes. For example:

```
let address = {
    'building no': 3960,
    street: 'North 1st street',
    state: 'CA',
    country: 'USA'
};
```

To access the `'building no'`, you must use the array-like notation:

```
address['building no'];
```

If you use the dot notation, you will get an error:

```
address.'building no';
```

Error:

```
SyntaxError: Unexpected string
```

Reading from a property that does not exist will result in an `undefined`. For example:

```
console.log(address.district);
```

Output:

```
undefined
```

Change the property's value

To change the value of a property, you use the assignment operator. For example:

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

person.firstName = 'Jane';

console.log(person);
```

Output:

```
{ firstName: 'Jane', lastName: 'Doe' }
```

Add a new property to an object

Unlike objects in other programming languages such as Java and C#, you can add a property to an object after creating it.

The following statement adds the `age` property to the `person` object and assigns 25 to it:

```
person.age = 25;
```

Delete a property of an object

To delete a property from an object, you use the `delete` operator:

```
delete objectName.propertyName;
```

The following example removes the `age` property from the `person` object:

```
delete person.age;
```

Check if a property exists

To check if a property exists in an object, you use the in operator:

```
propertyName in objectName
```

The following example creates an `employee` object and uses the `in` operator to check if the `ssn` and `employeeId` properties exist in the object.

```
let employee = {
    firstName: 'Peter',
    lastName: 'Doe',
```

```
    employeeId: 1
};

console.log('ssn' in employee);
console.log('employeeId' in employee);
```

Output:

```
false
true
```

Iterate over properties of an object using for...in loop

To iterate over all properties of an object without knowing property names, you use the `for...in` (https://www.javascripttutorial.net/javascript-for-in/) loop:

```
for(let key in object) {
    // ...
};
```

For example, the following statement creates a `website` object and iterates over its properties using the `for...in` loop:

```
let website = {
    title: 'JavaScript Tutorial',
    url: 'https://www.javascripttutorial.net',
    tags: ['es6', 'javascript', 'node.js']
};

for (const key in website) {
    console.log(website[key]);
}
```

Methods

Objects have actions. The actions are represented by **functions**. The following snippet adds the `greet` action to the `person` object:

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

person.greet = function () {
    console.log('Hello, World!');
}
person.greet();
```

Output:

```
Hello, World!
```

In this example, we added a function expression to create the function and assigned it to the property `greet` of the `person` object.

Then, we call the function via the `greet` property as `greet()`. When a function is a property of an object, it is called a method.

Besides using a function expression, you can define a function and add it to the object, like this:

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

function greet() {
    console.log('Hello, World!');
}

person.greet = greet;

person.greet();
```

Method shorthand

You can define methods using the object literal syntax:

```
let person = {
    firstName: 'John',
    lastName: 'Doe',
    greet: function () {
        console.log('Hello, World!');
    }
};
```

In ES6, you can even make it shorter:

```
let person = {
    firstName: 'John',
    lastName: 'Doe',
    greet() {
        console.log('Hello, World!');
    }
};

person.greet();
```

The `this` value

Typically, methods need to access data stored in the object.

For example, you may want to develop a method that returns the full name of the person object by concatenating the first name and last name.

Inside the method, the `this` value references the object that contains the method so you can access an object property using the dot notation:

```
this.propertyName
```

The following example uses the `this` value in the `getFullName()` method:

```
let person = {
    firstName: 'John',
    lastName: 'Doe',
    greet: function () {
        console.log('Hello, World!');
    },
    getFullName: function () {
        return this.firstName + ' ' + this.lastName;
    }
};


console.log(person.getFullName());
```

Output

John Doe

Summary

An object is a collection of key-value pairs called properties. A property key is a string and value can be any valid value.

Use the dot notation (`.`) or array-like notation (`[]`) to access an object property.

The `delete` operator removes a property from an object.

The `in` operator check if a property exists in an object.

The `for...in` iterates over properties of an object.

When functions are properties of an object, they are called methods.

Use the `this` inside the method to access the object's properties.