

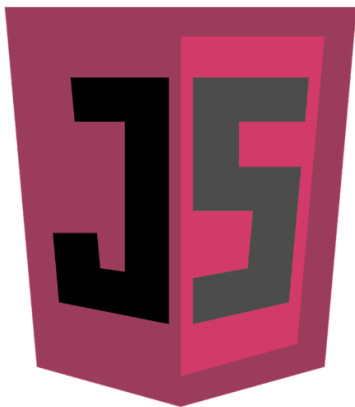
The JavaScript Object Notation

In this lesson, we cover the JavaScript object notation.
Let's begin!

WE'LL COVER THE FOLLOWING



- JSON syntax
- **Listing 8-1:** Using JSON in JavaScript



The Object Notation




In several code snippets, you already saw an interesting notation when displaying an object value. For example, take a look at this short code:

```
var car = new Object();  
car.manufacturer = "BMW";  
car.type = "X5";  
console.log(car);
```



When you run this code snippet, Chrome displays this console output:

When you run this code snippet, Chrome displays this console output.

 console

```
Object {manufacturer: "BMW", type: "X5"}
```



The notation you find between the braces, including the braces themselves, are a perfect description of the object. You can understand that this object contains two properties, `manufacturer` and `type`, and you also know the values of these properties are “BMW” and “X5”. This kind of notation is not an arbitrary one used by Chrome.

It is called JavaScript Object Notation (JSON). It is a versatile notation that, amongst other things, is used for displaying JavaScript objects. JSON is a data format widely used for data exchange among components of web applications, as an alternative of XML, and other presentation formats for transmitting structured data over the Internet.

As you will learn, JavaScript uses the JSON syntax, however, JSON is not a part of the ECMAScript specification.

 **NOTE:** The JSON notation format is described in RFC 4627.

JSON syntax

JSON provides a very simple syntax to describe structured information with a combination of simple values, objects, and arrays. Compared to XML, it contains less noise, and uses fewer characters to describe the same data.

Numbers, strings, Boolean values, and the null value are represented in JSON with the same syntax as in JavaScript, with the only exception that strings must be written between double quotes. So, these are all valid examples of single values in JSON:

```
123
0x35fa
"Hummingbird"
null
true
42.5
```



 **NOTE:** The undefined value is not supported in JSON.

Objects are represented with name and value pairs, where names are written between double quotes. The name and value are separated by a colon; properties are delimited by comma characters. The list of properties is wrapped in curly braces.

Here is a simple example:

```
{
  "manufacturer": "BMW",
  "type": "X5"
}
```



Compound objects can be described so that object values are represented with a nested JSON notation, just like in this sample:

```
{
  "id": 4234,
  "name": {
    "first": "John",
    "last": "Doe",
  },
  "enables": true
}
```



Collections of values and objects also can be represented by JSON, similarly to JavaScript arrays. The elements of a collection are wrapped between square brackets, individual elements are written with their native JSON representation, and elements are delimited with comma characters.

Here is a simple example:

```
[23, { "type": "S", "name": "Small" }, true]
```



Obviously, you can combine the value, object, and array notations to represent compound objects (arrays in object, objects in arrays, and so on) that describe large and complex structures.

JavaScript understands JSON natively. So, you can create an order in JavaScript, as shown in Listing 8-1.

Listing 8-1: Using JSON in JavaScript

```
<!DOCTYPE html>
<html>
<head>
  <title>Using JSON in JavaScript</title>
  <script>
    var order = {
      "date": "11/20/2013",
      "customerId": 116,
      "items": [
        {
          "product": "Surface 4 Pro",
          "unitprice": "799",
          "amount": 1
        },
        {
          "product": "Type Cover 4",
          "unitprice": "129",
          "amount": 1
        },
        {
          "product": "Docking station",
          "unitprice": "199",
          "amount": 1
        }
      ]
    }
    console.log("Date: " + order.date);
    for (var i = 0; i < order.items.length; i++) {
      console.log("Product: "
        + order.items[i].product);
      console.log("Unit price: "
        + order.items[i].unitprice);
      console.log("Amount: "
        + order.items[i].amount);
    }
  </script>
</head>
<body>
  Listing 8-1: View the console output
</body>
</html>
```

 **NOTE:** When using JSON in JavaScript to initialize objects, you can omit the double quotes from property names.

Here, the order object is defined with JSON. After initializing the order variable, you can access its properties just as if it were created with a constructor setting up its properties with imperative code. Not surprisingly, the output of this code looks like this:

JS console

```
Date: 11/20/2013
Product: Surface 4 Pro
Unit price: 799
Amount: 1
Product: Type Cover 4
Unit price: 129
Amount: 1
Product: Docking station
Unit price: 199
Amount: 1
```

In the *next lesson* we'll explore how to serialize and parse a JSON document.

See you there! :)