Working with JSON in JavaScript

**A user guide on how to work with JSON in Javascript.**

Introduction

JSON, short for JavaScript Object Notation, is a language-independent data format that has been accepted as an industry standard. Because it is based on the JavaScript programming language, JSON’s syntax looks similar to a JavaScript object with [minor differences](https://www.codecademy.com/articles/what-is-json). We’ll take a look at the subtle difference between them. Later on, we’ll learn how to parse JSON and extract its content as JavaScript. Lastly, we’ll learn how to write a JSON object with JavaScript. So, let’s begin.

JSON Object vs. JavaScript Object

Here is an example JSON object of a person named Kate, who is 30 years old, and whose hobbies include reading, writing, cooking, and playing tennis:

{  
  "person": {    
    "name": "Kate",    
    "age": 30,    
    "hobbies": [ "reading", "writing", "cooking", "tennis" ]   
  }  
}

Represented as a JavaScript object literal, the same information would appear as:

{    
  person: {  
    name: 'Kate',    
    age: 30,    
    hobbies: [ 'reading', 'writing', 'cooking', 'tennis' ]   
  }  
}

Notice a slight difference between the two formats.

* The name portion in each JSON name-value pair and all string values must be enclosed in double-quotes while this is optional in JavaScript.
* JavaScript accepts string values that are single or double-quoted, however, there exists JavaScript [coding guidelines](https://en.wikipedia.org/wiki/Programming_style) that prefer one style over another.

Reading a JSON String

In a typical web application, the JSON data that we receive from a web request comes in the form of a string. At other times, JSON data is stored in a file that is used for authentication, configuration, or database storage. These files typically have a .json extension, and they have to be opened in order to retrieve the JSON string in it. In either case, we will need to convert the string to a format that we can use in order to access its parts. Each programming language has its own mechanism to handle this conversion. In JavaScript, for example, we have a built-in JSON class with a method called [**.parse()**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse) that takes a JSON string as a parameter and returns a JavaScript object.

The following code converts a JSON string object, **jsonData**, into a JavaScript object, **jsObject**, and logs **jsObject** on the console.

const jsonData = '{ "book": { "name": "JSON Primer", "price": 29.99, "inStock": true, "rating": null } }';  
   
const jsObject = JSON.parse(jsonData);  
   
console.log(jsObject);

This will print out **jsObject** as follows:

{  
  book: { name: 'JSON Primer', price: 29.99, inStock: true, rating: null }  
}

Once we have converted a JSON object to a JavaScript object, we can access the individual properties inside the JavaScript object. To access a value inside a JavaScript object based on its property name, we can either use dot notation, (**.propertyName**), or bracket notation, (**['propertyName']**).

For instance, to retrieve the **book** property of **jsObject** we could do the following:

// Using the dot notation  
const book = jsObject.book;      
console.log(book);  
console.log(book.name, book.price, book.inStock);  
   
// Using the bracket notation  
const book2 = jsObject['book'];  
console.log(book2);  
console.log(book2["name"], book2["price"], book2["inStock"]);

Both ways of accessing the book property return the same output:

{ name: 'JSON Primer', price: 29.99, inStock: true, rating: null }  
JSON Primer 29.99 true

As you can see, after parsing **jsonData** into a JavaScript object that’s stored in the variable, **book**, you can treat **book** like any other object! That means you can access property values, as shown above, edit existing values, iterate over the keys and values, etc…

Exercise: Reading a JSON String

Now that we’ve shown you how to read a JSON string, let’s practice with a code challenge by writing some code yourself.

Code Challenge

1. Create a variable called **jsObject** that is an object parsed from **jsonData**.
2. Print out the array of all the **children** property nested in **jsObject**. Be sure to use either bracket notation or dot notation to access the nested properties.

Check Answer

Writing a JSON String

Before we can send off our data across the web, we need to convert them to a JSON string. In JavaScript, we would use the built-in **JSON** class method, [**.stringify()**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify) to transform our JavaScript object to a JSON string.

The following code converts a JavaScript object, **jsObject**, into a JSON string, **jsonData**.

const jsObject = { book: 'JSON Primer', price: 29.99, inStock: true, rating: null };  
const jsonData = JSON.stringify(jsObject);  
console.log(jsonData);

This will display the following output:

{ "book": "JSON Primer", "price": 29.99, "inStock": true, "rating": null }

Exercise: Writing a JSON String

Now that we’ve learned how to convert our JavaScript object to a JSON string, let’s practice with another code challenge by you writing some code.

Code Challenge

As a developer, you receive some data in the form of a JSON string in the variable, **jsonData**. However, the content of **jsonData** is not completely correct. The **age** value of the parent property should be **35** instead of **45**. Without changing the content of **jsonData** directly, update the **age** value and then log a new JSON string with the correct value in the console.

Here is a step-by-step guide to solve this challenge:

1. Convert **jsonData** to a JavaScript object using **JSON.parse()** and save it as a **const** variable, for instance, **jsObject**.
2. Use either the dot, **.key**, or bracket, **['key']**, notation to access the **parent** property of **jsObject** followed by the **age** property and change its value from **45** to **35**.
3. Convert **jsObject** back to a JSON string using **JSON.stringify()** and save it as another const variable, for instance, **jsObjectToJson**.
4. Log the **jsObjectToJson** string on the console.

1

2

3

4

5

6

7

8

const jsonData = '{"parent":

{"name":"Sally","age":45,"children":[

{"name":"Kim","age":3},{"name":"Lee",

"age":1}]}}';

const jsObject = JSON.parse(jsonData);

jsObject.parent.age=35;

console.log(jsObject)

const newJsData=JSON.stringify(jsObject)

console.log(newJsData)



Run

Output:



Check Answer

Review

In this article, you have learned how to do the following:

* Compare JSON with JavaScript’s Object literal syntax.
* Convert a JSON string into a JavaScript Object.
* Convert a JavaScript Object into a JSON string.

Congratulations on reaching this milestone! With a little more practice, you will soon handle JSON in Javascript with ease.

**Working with JSON in JavaScript**

**A user guide on how to work with JSON in Javascript.**

**Introduction**

JSON, short for JavaScript Object Notation, is a language-independent data format that has been accepted as an industry standard. Because it is based on the JavaScript programming language, JSON’s syntax looks similar to a JavaScript object with [minor differences](https://www.codecademy.com/articles/what-is-json). We’ll take a look at the subtle difference between them. Later on, we’ll learn how to parse JSON and extract its content as JavaScript. Lastly, we’ll learn how to write a JSON object with JavaScript. So, let’s begin.

**JSON Object vs. JavaScript Object**

Here is an example JSON object of a person named Kate, who is 30 years old, and whose hobbies include reading, writing, cooking, and playing tennis:

{  
  "person": {    
    "name": "Kate",    
    "age": 30,    
    "hobbies": [ "reading", "writing", "cooking", "tennis" ]   
  }  
}

Represented as a JavaScript object literal, the same information would appear as:

{    
  person: {  
    name: 'Kate',    
    age: 30,    
    hobbies: [ 'reading', 'writing', 'cooking', 'tennis' ]   
  }  
}

Notice a slight difference between the two formats.

* The name portion in each JSON name-value pair and all string values must be enclosed in double-quotes while this is optional in JavaScript.
* JavaScript accepts string values that are single or double-quoted, however, there exists JavaScript [coding guidelines](https://en.wikipedia.org/wiki/Programming_style) that prefer one style over another.

**Reading a JSON String**

In a typical web application, the JSON data that we receive from a web request comes in the form of a string. At other times, JSON data is stored in a file that is used for authentication, configuration, or database storage. These files typically have a **.json** extension, and they have to be opened in order to retrieve the JSON string in it. In either case, we will need to convert the string to a format that we can use in order to access its parts. Each programming language has its own mechanism to handle this conversion. In JavaScript, for example, we have a built-in JSON class with a method called [**.parse()**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse) that takes a JSON string as a parameter and returns a JavaScript object.

The following code converts a JSON string object, **jsonData**, into a JavaScript object, **jsObject**, and logs **jsObject** on the console.

const jsonData = '{ "book": { "name": "JSON Primer", "price": 29.99, "inStock": true, "rating": null } }';  
   
const jsObject = JSON.parse(jsonData);  
   
console.log(jsObject);

This will print out **jsObject** as follows:

{  
  book: { name: 'JSON Primer', price: 29.99, inStock: true, rating: null }  
}

Once we have converted a JSON object to a JavaScript object, we can access the individual properties inside the JavaScript object. To access a value inside a JavaScript object based on its property name, we can either use dot notation, (**.propertyName**), or bracket notation, (**['propertyName']**).

For instance, to retrieve the **book** property of **jsObject** we could do the following:

// Using the dot notation  
const book = jsObject.book;      
console.log(book);  
console.log(book.name, book.price, book.inStock);  
   
// Using the bracket notation  
const book2 = jsObject['book'];  
console.log(book2);  
console.log(book2["name"], book2["price"], book2["inStock"]);

Both ways of accessing the book property return the same output:

{ name: 'JSON Primer', price: 29.99, inStock: true, rating: null }  
JSON Primer 29.99 true

As you can see, after parsing **jsonData** into a JavaScript object that’s stored in the variable, **book**, you can treat **book** like any other object! That means you can access property values, as shown above, edit existing values, iterate over the keys and values, etc…

**Exercise: Reading a JSON String**

Now that we’ve shown you how to read a JSON string, let’s practice with a code challenge by writing some code yourself.

Code Challenge

1. Create a variable called **jsObject** that is an object parsed from **jsonData**.
2. Print out the array of all the **children** property nested in **jsObject**. Be sure to use either bracket notation or dot notation to access the nested properties.

Check Answer

**Writing a JSON String**

Before we can send off our data across the web, we need to convert them to a JSON string. In JavaScript, we would use the built-in **JSON** class method, [**.stringify()**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify) to transform our JavaScript object to a JSON string.

The following code converts a JavaScript object, **jsObject**, into a JSON string, **jsonData**.

const jsObject = { book: 'JSON Primer', price: 29.99, inStock: true, rating: null };  
const jsonData = JSON.stringify(jsObject);  
console.log(jsonData);

This will display the following output:

{ "book": "JSON Primer", "price": 29.99, "inStock": true, "rating": null }

**Exercise: Writing a JSON String**

Now that we’ve learned how to convert our JavaScript object to a JSON string, let’s practice with another code challenge by you writing some code.

Code Challenge

As a developer, you receive some data in the form of a JSON string in the variable, **jsonData**. However, the content of **jsonData** is not completely correct. The **age** value of the parent property should be **35** instead of **45**. Without changing the content of **jsonData** directly, update the **age** value and then log a new JSON string with the correct value in the console.

Here is a step-by-step guide to solve this challenge:

1. Convert **jsonData** to a JavaScript object using **JSON.parse()** and save it as a **const** variable, for instance, **jsObject**.
2. Use either the dot, **.key**, or bracket, **['key']**, notation to access the **parent** property of **jsObject** followed by the **age** property and change its value from **45** to **35**.
3. Convert **jsObject** back to a JSON string using **JSON.stringify()** and save it as another const variable, for instance, **jsObjectToJson**.
4. Log the **jsObjectToJson** string on the console.

1

2

3

4

5

6

7

8

const jsonData = '{"parent":

{"name":"Sally","age":45,"children":[

{"name":"Kim","age":3},{"name":"Lee",

"age":1}]}}';

const jsObject = JSON.parse(jsonData);

jsObject.parent.age=35;

console.log(jsObject)

const newJsData=JSON.stringify(jsObject)

console.log(newJsData)



Run

Output:



Check Answer

**Review**

In this article, you have learned how to do the following:

* Compare JSON with JavaScript’s Object literal syntax.
* Convert a JSON string into a JavaScript Object.
* Convert a JavaScript Object into a JSON string.

Congratulations on reaching this milestone! With a little more practice, you will soon handle JSON in Javascript with ease.