W02 Learning Activity: JSON

Overview

Whenever information is passed from one program to another or saved to disk, the data needs to be formatted in a way that it can be understood. JSON is a very popular format for storing data and almost all websites use it in some form because it is simple, easy to use, and very flexible.

In this learning activity you will learn about the JSON format and how to create a JSON file to be used later.

JSON is an important topic that you will use throughout the Software Development program.

Prepare

JSON stands for **JavaScript Object Notation** and is a simple data interchange format. JSON is often used for transmitting data between a server and a client application and can be used for configuration files and data storage.

JSON is a text-based format that is easy for people and computers to read and understand, and it can be used by any program language.

In JSON, you store lists (or more precisely, dictionaries) of key-value pairs in the format `key:value`, such as `"name": "John"`. When you want to store multiple key-value pairs, you enclose them in curly braces `{ }` and separate them with commas like this:

```
{
  "firstName": "John",
  "lastName": "Taylor",
  "age": 34
}
```

Whitespace such as new lines and indentation is not required, but it is helpful for people that may look at the data.

Also, you may notice from this example that strings are specified with quotation marks **" "**, but quotation marks are not used for numbers, or for the boolean values **true** and **false**.

Square brackets **[ ]** are used to create an array or list of values, rather than key-value pairs.

In addition to the simple structure shown above, the values themselves can be dictionaries that form nested structures. The following shows a more complex example that includes a list of values for **"courses"** using square brackets **[ ]** and a nested structure for the **"address"**:

```
{
  "name": "Sariah Lucias",
  "age": 27,
  "isCurrentStudent": true,
  "courses": ["WDD231", "CSE210", "REL340"],
  "address": {
    "street": "123 Bedford Road",
    "city": "Gweru",
    "state": "Midlands Province",
    "country": "Zimbabwe",
    "zip": "00000"
  }
}
```

Note that the example above is a valid JSON object with key-value pairs. The keys are enclosed in double quotes, and the key-value pairs are separated by commas. It contains a mix of data types, including strings ("Sariah Lucias"), numbers (27), Booleans (true), arrays (["WDD231", "CSE210", "REL340"]), and nested objects (address).

The following are some key points about JSON:

- JSON is a lightweight, text-based, data interchange format that is easy for humans to read and write and easy for machines to parse and generate.
- JSON is language-independent, meaning it can be used with any programming language that supports text-based data formats.
- JSON is often used in web applications to exchange data between a client and a server.
- JSON is closely aligned with the way JavaScript objects are stored, so it can easily be converted to a native JavaScript object.
- JSON is written using basic key/value pairs in this format: **key:value** and supports data types of strings, numbers, arrays, Booleans, and other object literals. There are no methods.

- JSON requires double quotes to be used around string and property names and follows strict adherence to comma or colon placement.
- JSON does not contain nor allow comments.
- JSON files are typically stored in a file that uses the **.json** file extension.

Using JSON data in JavaScript

Consuming and using JSON data in JavaScript is a fundamental skill, especially when working with APIs or storing structured data.

You can create a native JavaScript object directly, or you can load it from a JSON string. The following shows how to create an object directly (not using JSON):

```
const studentObject1 = {
  name: "John",
  age: 25,
  isStudent: true
};
```

Notice that this looks very similar to the JSON string format above, but it is slightly different. For example, the keys do not have quotation marks.

To create an object from a string of JSON data, use the built-in **`JSON.parse()`** method. This takes the JSON string and converts it into a JavaScript object. It is useful for parsing JSON data received from a server or local file.

```
const jsonString = '{"name": "John", "age": 25, "isStudent": true}';

const studentObject2 = JSON.parse(jsonString);
```

To convert a native JavaScript object into a JSON string, use the built-in **`JSON.stringify()`** method. This method takes a JavaScript object and converts it into a JSON string. It is useful for sending data to a server or saving it in a file.

```
const studentJsonString = JSON.stringify(studentObject1);
```

> To load JSON string data from a server or a local file, you use the **`fetch()`** method. This will be explained in detail in another learning activity.

Activity Instructions

For this activity you will create a new JSON file and make sure that it is valid.

Step 1: Identify the Structure

1. Open this example JSON file in a new tab. (Note that this file does not contain newlines and indentation, and yet, most browsers will format it nicely to make the data more readable.)
2. Identify some of the **key-value** pairs and **data types** used in this example.

> Here are some example keys (property names) found in this JSON file.
>
> - **key**: "location", **value**: "Delphi", **data type**: string
> - **key**: "id", **value**: 89.99, **data type**: number
> - **key**: "inStock", **value**: true, **data type**: Boolean
> - **key**: "products", **value**: (there are three values), **data type**: an array of objects and each has five key-value pairs (properties)

Step 2: Create a JSON File

1. Create a JSON file that will hold new ward member information.
2. Populate this new JSON file with valid syntax using the following data requirements:
    1. family name
    2. the date the family moved into the ward
    3. the number of people in the family (number of records)
    4. if visited by bishopric?
    5. individual family member data
        1. name
        2. gender
        3. birthdate
3. Test your JSON file in the browser and validate it using a tool like JSONLint or another JSON validator/formatter.

▼ **Check Your Understanding (example solution)**

```
{
  "family_name": "Santos",
  "move_in_date": "2024-06-24",
  "number_of_people": 4,
  "visited_by_bishopric": true,
  "family_members": [
    {
```

```
      "name": "Pedro Santos",
      "gender": "Male",
      "birthdate": "1990-05-15"
    },
    {
      "name": "Maria Santos",
      "gender": "Female",
      "birthdate": "1992-08-20"
    },
    {
      "name": "Andrea Santos",
      "gender": "Female",
      "birthdate": "2010-03-05"
    },
    {
      "name": "Luz Santos",
      "gender": "Male",
      "birthdate": "2012-11-10"
    }
  ]
}
```

Optional Resources

- JSON – MDN