Appendix D. Using JSON formatting

In this appendix, we discuss JavaScript Object Notation (JSON). JSON is an often-used way to format the data exchanged by apps in the HTTP request and response when using REST endpoints to communicate (figure D.1). Because REST endpoints are one of the most encountered ways to establish communication between apps, and JSON is the main way to format the data exchanged, understanding and knowing how to use JSON formatting is essential.

Figure D.1 When you implement business logic, it sometimes implies establishing communication between multiple apps. Most often, you use JSON to format the data the apps exchange. To implement and test your REST endpoints, you need to understand JSON.

Fortunately, JSON is easy to understand, and it only follows a few rules. First, you need to know that what you represent with JSON are object instances using their attributes. Like in the case of a Java class, the attributes are identified with names and hold values. You may say the object `Product` has an attribute `name` and an attribute `price`. An instance of the `Product` class assigns values to the attributes. For example, the `name` is "chocolate" and the `price` is 5. If you want to represent this in JSON, you need to consider the following rules:

- To define an object instance in JSON, we use curly braces.
- Between the curly braces, we enumerate the attribute-value pairs, separating them with commas.
- The attribute names are written between double quotes.
- The string values are written between double quotes. (Any double quote the string contains needs to have a backslash "\" in front of it.)
- The numerical values are written without quotes.
- The attribute name and its value are separated by a colon.

Figure D.2 presents the JSON-formatted product instance with the attribute name "chocolate" and the price "5."

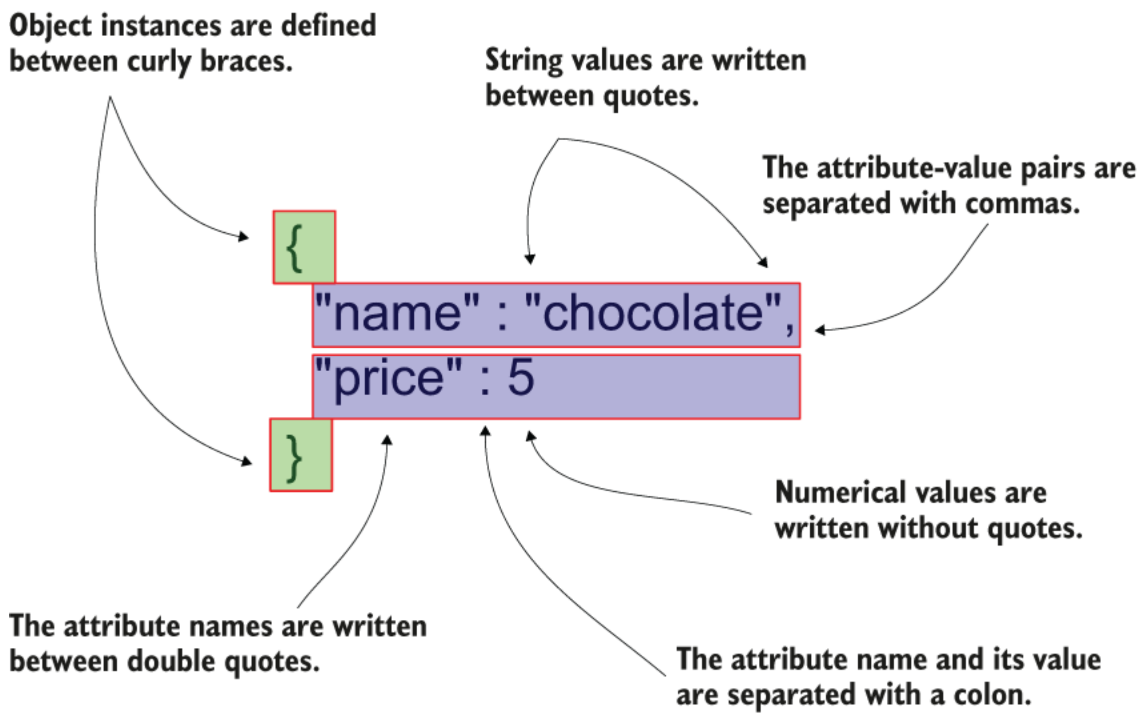
Object instances are defined between curly braces.
String values are written between quotes.
The attribute-value pairs are separated with commas.
{
"name" : "chocolate",
"price" : 5
}
Numerical values are written without quotes.
The attribute names are written between double quotes.
The attribute name and its value are separated with a colon.

Figure D.2 Describing an object instance in JSON. We surround the attribute-value pairs with curly braces. A colon separates the attribute name and its value. The attribute-value pairs are separated with commas.

In JSON, the object itself doesn't have a name or a type. Nowhere does it say the snippet describes a product. The object's only relevant items are its attributes. Figure D.2 details the JSON rules for describing an object.

An object might contain another object instance as the value of one of its attributes. If the `Product` has a `Pack` and the `Pack` is an object described by its attribute `color`, then a representation of a `Product` instance looks like the next snippet:

```
{
   "name" : "chocolate",
   "price" : 5,
   "pack" : {               ❶
       "color" : "blue"
   }
}
```

❶ The value of the attribute pack is an object instance.

The same rules repeat. You can have many attributes representing other objects and nest them as many times as you need.

If you want to define a collection of objects in JSON, you use brackets, and you separate the entries with commas. The next code snippet shows you how to define a collection that contains two `Product` instances:

```
[                             ❶
  {
    "name" : "chocolate",
    "price" : 5
  },                          ❷
  {
    "name" : "candy",
    "price" : 3
  }
]
```

❶ We use brackets to surround the object instances in a collection.

❷ The instances are separated with commas.