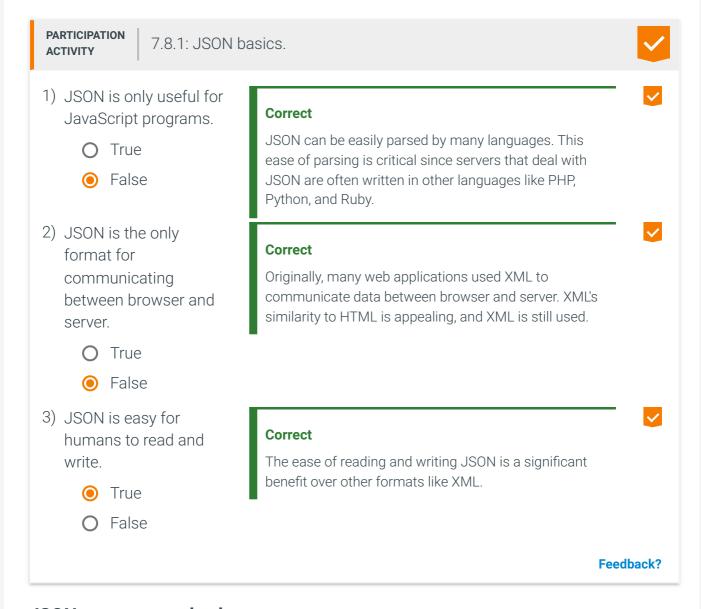
# 7.8 JavaScript Object Notation (JSON)

#### Introduction to JSON

Communicating data between the server and browser is a significant task for modern web applications. Initial attempts to do so included unstructured text documents and heavily structured XML documents, both of which required significant effort to convert to a usable format. *JavaScript Object Notation*, or *JSON*, is an efficient, structured format for data based on a subset of the JavaScript language. JSON (pronounced "Jason") is intended to be easily readable by humans and computers. Debugging communication that uses JSON is easy because humans can read JSON. Communication is efficient because computers can transmit and parse JSON quickly. As a result, JSON has rapidly become the dominant format of data transfer between web browsers and servers.



## **JSON** structure and values

JSON has six basic data types:

- String Unicode characters enclosed within double quotes ("). A few special characters must be escaped with a backslash (\). Ex: backslashes (\\), double quotes (\"), newlines (\n), and tabs (\t).
- 2. **Number** Either an integer or decimal number. Ex: 42, 3.141, -1.1e-5.
- 3. **Object** Unordered list of zero or more name/value pairs separated by commas and enclosed within braces ({}). A name in a JSON object must be a string in double quotes. A value can be any legal JSON value. Each name and value is separated by a colon. Ex: { "Name": "Joe", "Age": 35 }
- 4. **Array** Ordered list of zero or more JSON values separated by commas and enclosed within brackets ([]). Ex: [] and [13, "blue"].
- 5. **Boolean** Either true or false.
- 6. **null** Represents "nothing".

A **JSON value** can be any of the above data types.

The JSON structure is defined recursively so that objects can contain arrays and arrays can contain objects to any arbitrary depth.

A common error when generating JSON programmatically is to include a trailing comma after the list of name/value pairs in a JSON object or after the list of JSON values in a JSON array. Ex: [0, 1, 2,].

Figure 7.8.1: An example JSON data structure.

```
{
   "name": "John Doe",
   "vehicles": [
         "make": "Ford",
         "model": "F-150",
         "color": "white"
      },
         "make": "Toyota",
         "model": "Camry",
         "color": "red"
   ],
   "married": false,
   "previous_customer":
true,
   "known_associates": [],
   "notes": null
```

Feedback?

The JSON structure above is an object with six name/value pairs:

- 1. name has the string value John Doe.
- 2. **vehicles** has an array value of two objects. Each object in the vehicles array has three name/value pairs: make, model, and color.
  - 1. The array's first object's make is the string Ford, model is the string F-150, and color is the string white.
  - 2. The array's second object's make is Toyota, model is Camry, and color is red.
- 3. married is false.
- 4. previous customer is true.
- 5. known\_associates is an empty array.
- 6. notes is null.

PARTICIPATION ACTIVITY

7.8.2: JSON data types.



Refer to the following JSON structure:

1) What value type does the JSON structure create?



O object

String

#### **Correct**

The JSON structure produces an array because the rest of the internal values are surrounded by brackets.

<ol><li>How many objects does the JSON</li></ol>	Correct
structure create?	JSON objects are wrapped in braces. The array created
0 1	by the JSON structure contains three objects.
<b>③</b> 3	
O 4	
3) What is the data type of favorite?	Correct
array	favorite only contains true or false values.
boolean	
String	
4) What is the data type of <b>created</b> ?	Correct
number	created only contains digits for values.
O object	
String	
	Feedback?

## **Working with JSON**

JavaScript provides a built-in **JSON object** that provides two methods for working with JSON:

- 1. The **JSON.parse()** method creates a JavaScript object from a string containing JSON. Ex: **JSON.parse('[1,"two",null]')** converts the string '[1,"two",null]'into the JavaScript array [1,"two",null]. Typically, **JSON.parse()** is used with data received from a server.
- 2. The **JSON.stringify()** method creates a string from a JavaScript object. Typically, **JSON.stringify()** is used with data sent to a server. **JSON.stringify()** creates a string representation of any passed object by either calling the object's **toJSON()** method if defined or recursively serializing all enumerable, non-function properties. Ex: **JSON.stringify(new Date('2020-08-06'))** converts the JavaScript Date object to the string **2020-08-06T00:00:00.000z** by calling the Date object's **toJSON()** method.

Good practice is to use single quotes around JavaScript strings containing JSON notation so that the double quotes for strings and JSON object names do not need to be escaped. Ex: Use '{"name":"Bob"}' instead of "{\"name\":\"Bob\"}".

PARTICIPATION ACTIVITY

7.8.3: JSON.parse and JSON.stringify example.



```
let bondStr = '{"name":"James", "age":35}';
console.log(bondStr);

let bondObj = JSON.parse(bondStr);
console.log("Happy birthday, " + bondObj.name);

bondObj.age += 1;
bondStr = JSON.stringify(bondObj);
console.log(bondStr);

{"name":"James", "age":35}

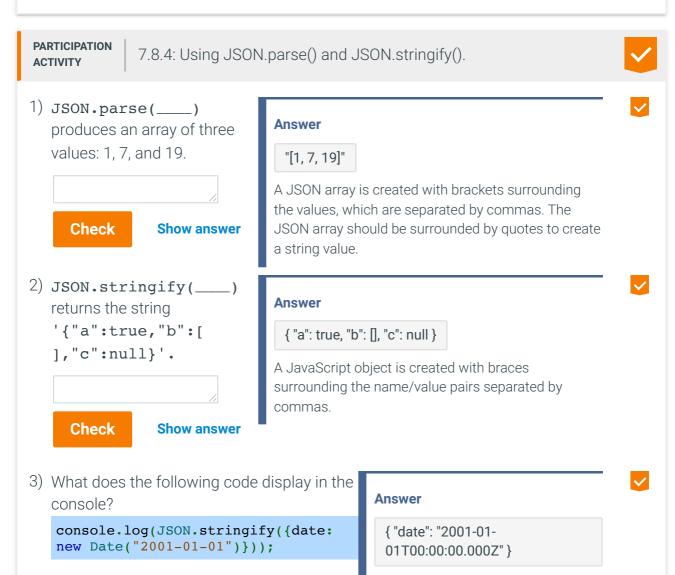
{"name":"James", "age":36}
```

bondObj's age property is incremented. The JSON.stringify() method then converts bondObj back to a JSON string.

## Captions ^

- 1. bondStr is a string representing a JSON object.
- 2. The JSON.parse() method parses the JSON string to create a JavaScript object. The JavaScript object's name property is then printed to the console.
- 3. bondObj's age property is incremented. The JSON.stringify() method then converts bondObj back to a JSON string.

Feedback?





## **Extending and customizing JSON output**

The JSON.parse() method's second parameter is an optional parameter for a reviver function. A **reviver function** is used to modify parsed values before being returned, and is helpful when a JSON string represents a data type not available in JSON. Ex: A reviver function can convert a string representing a date, "2010–12–30", to a JavaScript Date object.

The JSON.stringify() method has two optional parameters: a replacer and a spacer. The replacer enables customization of the generated string. If replacer is a function, JSON.stringify() will use the value returned by the function as the string representation. Ex: A replacer can convert a JavaScript type not directly supported in JSON to a string representation of that data type. If replacer is an array, JSON.stringify() will filter the returned value by converting only the properties listed in the replacer array. Ex: JSON.stringify({a:1,b:2,c:3},["a","b"]) returns the string '{"a":1,"b":2}'.

The spacer controls the indentation spacing of output JSON string, which indicates the depth of values in the object. When the spacer parameter is specified and not an empty string, the output will also include newlines. Ex: JSON.stringify({a:1,b:2}, null, " ") returns the string below because the spacer parameter is a string with two spaces.

```
'{
    "a": 1,
    "b": 2
}'
```

```
PARTICIPATION
              7.8.5: Reviver function for JSON.parse().
ACTIVITY
     1 2 3 4 < 2x speed
  let data = { date:new Date("2010-10-10") };
                                                   Object {date: Sat Oct 09 2010 20:00:0
  console.log(data);
                                                   GMT-0400 (EDT)}
  let json = JSON.stringify(data);
                                                   {"date": "2010-10-10T00:00:00.000Z"}
  console.log(json);
                                                   Object {date: "2010-10-10T00:00:00.00
  console.log(JSON.parse(json));
                                                   Object {date: Sat Oct 09 2010 20:00:0
  console.log(JSON.parse(json, function(k,v) {
                                                   GMT-0400 (EDT)}
    if (k == "date") return new Date(v);
     return v;
  }));
```

By providing a reviver function, JSON.parse() converts the date string to a Date object.

#### Captions ^

- 1. The console displays the date property of the data JavaScript object to be a Date object.
- 2. JSON.stringify() converts the Date object to a string.
- 3. JSON.parse() converts the string in json to a JavaScript string.
- 4. By providing a reviver function, JSON.parse() converts the date string to a Date object.

Feedback?

## PARTICIPATION ACTIVITY

7.8.6: Customizing JSON.parse and JSON.stringify.



 Which optional parameter can convert the string representation of a date into a JavaScript Date object?

#### **Correct**

The optional reviver parameter takes a key and value pair and returns a JavaScript representation of the value.

- replacer
- reviver
- Spacer
- 2) What is the result of the following JSON.stringify() call?

```
JSON.stringify({a:"one",b:"two",c:"three"},
["a","c"])
```

#### Correct

When the replacer parameter is an array, the JSON.stringify method only populates the final string with keys matching values in the array.



3) What is the result of the following JSON.stringify() call that uses two spaces for the space parameter?

#### Correct

**~** 

The spacer parameter controls the whitespace indentation of the output string.

```
JSON.stringify({a:
{b:1,c:3}}, null, '
')
```

- O '{"a": {"b":1,"c":3}}'
- '{ "a":{
   "b":1, "c":3
  }}'
- "a": {
   "b": 1,
   "c": 3
  }
- "a": {
   "b": 1,
   "c": 3,
   }
  }

Feedback?



7.8.1: JavaScript and JSON.



330096 4000608 ax3zav7

#### Start

