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.

PARTICIPATION ACTIVITY 7.8.1: JSON basics.

1) JSON is only useful for JavaScript programs.
   - ○ True
   - ◉ False

   Correct
   JSON can be easily parsed by many languages. This ease of parsing is critical since servers that deal with JSON are often written in other languages like PHP, Python, and Ruby.

2) JSON is the only format for communicating between browser and server.
   - ○ True
   - ◉ False

   Correct
   Originally, many web applications used XML to communicate data between browser and server. XML's similarity to HTML is appealing, and XML is still used.

3) JSON is easy for humans to read and write.
   - ◉ True
   - ○ False

   Correct
   The ease of reading and writing JSON is a significant benefit over other formats like XML.

Feedback?

JSON structure and values

JSON has six basic data types:

1. **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:

```
[
  { "name": "oreo",
    "type": "cookie",
    "flavors": ["chocolate", "vanilla"],
    "favorite": false,
    "created": 1912
  },
  { "name": "snickers",
    "type": "candy bar",
    "flavors": ["chocolate", "peanuts", "caramel", "nougat"],
    "favorite": true,
    "created": 1930
  },
  { "name": "malt",
    "type": "frozen dairy",
    "flavors": ["vanilla", "chocolate", "strawberry"],
    "favorite": false,
    "created": 1922
  }
]
```

1) What value type does the JSON structure create?

- (•) array
- ( ) object
- ( ) string

Correct

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

2) How many objects does the JSON structure create?

- 1
- 3
- 4

Correct

JSON objects are wrapped in braces. The array created by the JSON structure contains three objects.

3) What is the data type of `favorite`?

- array
- boolean
- string

Correct

`favorite` only contains true or false values.

4) What is the data type of `created`?

- number
- object
- string

Correct

`created` only contains digits for values.

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.

1 2 3 2x speed

```
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}
Happy birthday, James
{"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?

PARTICIPATION ACTIVITY 7.8.4: Using JSON.parse() and JSON.stringify().

1) `JSON.parse(____)` produces an array of three values: 1, 7, and 19.

Check Show answer

Answer

"[1, 7, 19]"

A JSON array is created with brackets surrounding the values, which are separated by commas. The JSON array should be surrounded by quotes to create a string value.

2) `JSON.stringify(____)` returns the string `'{"a":true,"b":[],"c":null}'`.

Check Show answer

Answer

{ "a": true, "b": [], "c": null }

A JavaScript object is created with braces surrounding the name/value pairs separated by commas.

3) What does the following code display in the console?

```
console.log(JSON.stringify({date:
new Date("2001-01-01")}));
```

Answer

{ "date": "2001-01-01T00:00:00.000Z" }

Check Show answer

`JSON.stringify()` converts the JavaScript Date object to a string, which is then displayed in the console.

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 ACTIVITY 7.8.5: Reviver function for JSON.parse().

1 2 3 4 2x speed

```
let data = { date:new Date("2010-10-10") };
console.log(data);

let json = JSON.stringify(data);
console.log(json);

console.log(JSON.parse(json));

console.log(JSON.parse(json, function(k,v) {
   if (k == "date") return new Date(v);
   return v;
}));
```

```
Object {date: Sat Oct 09 2010 20:00:0
GMT-0400 (EDT)}

{"date":"2010-10-10T00:00:00.000Z"}

Object {date: "2010-10-10T00:00:00.00

Object {date: Sat Oct 09 2010 20:00:0
GMT-0400 (EDT)}
```

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.

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

- replacer
- reviver
- spacer

Correct

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

2) What is the result of the following JSON.stringify() call?

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

- `'{"a":"one","c":"three"}'`
- `'{"a":"one","b":"two","c":"three"}'`
- `'{"b":"two"}'`

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?

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

- `'{"a":{"b":1,"c":3}}'`
- `'{  "a":{\n"b":1,    "c":3\n}}'`
- `'{\n  "a": {\n    "b": 1,\n    "c": 3\n  }\n}'` (selected)
- `'{\n    "a": {\n        "b": 1,\n        "c": 3,\n    }\n}'`

Correct

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

Feedback?

CHALLENGE ACTIVITY 7.8.1: JavaScript and JSON.

530096.4000608.qx3zqy7

Start

Assign jsonData with a JSON object with properties: studentName (a string), userAge (a number) and siblings (an array of strings). Note: The content of the properties doesn't matter.

```
1 let jsonData;
2 /* Your solution goes here */
3
```

1 2 3 4 5

Check Next

View your last submission

Feedback?

Exploring further:

- JSON from MDN

How was this section? Provide section feedback