Hyperiondev

TASK

Javascript VI: JSON

Visit our website

Introduction

WELCOME TO THE SIXTH JAVASCRIPT TASK!

You will find that JavaScript objects and arrays are used when creating many web applications! These are essential data structures that are used very often in web development. The fact that these data structures need to be used with HTTP when doing web development means that we need to use two very important tools: the Web Storage API and JSON. Both of these tools are introduced in this task.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our team is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

SENDING OBJECTS BETWEEN A WEB SERVER AND CLIENTS

HTTP (HyperText Transfer Protocol) is the protocol that allows for information to be transferred across the internet. Everything that is transferred over the web is transferred using HTTP. There are 2 very important facts about HTTP that we need to keep in mind though:

1. **HTTP is a stateless protocol.** That means HTTP doesn't see any link between two requests being successively carried out on the same connection. Cookies and the Web Storage API are used to store necessary state information

2. **HTTP transfers text (not objects or other complex data structures).** JSON converts data structures, like objects, into text that can be transferred using HTTP. HTTP can also be used to transfer other media types and not only text.

For some more information on the HTTP protocol please read this resource.

THE WEB STORAGE API: SESSIONSTORAGE

Thus far we have used variables to store data that is used in our programs. When storing data that is used for web applications, it is important to keep in mind that HTTP is a *stateless protocol*. That basically means that the web server doesn't store information about each user's interaction with the website. For example, if 100 people are shopping online, the webserver that hosts the online shopping application doesn't necessarily store the state of each person's shopping experience (e.g. how many items each person has added or removed from their shopping cart). Instead, that type of information is stored on the browser using **cookies** or the *Web Storage API* (the more modern and efficient solution for client storage). The Web Storage API stores data using key-value pairs. This mechanism of storing data has to a large extent replaced the use of cookies.

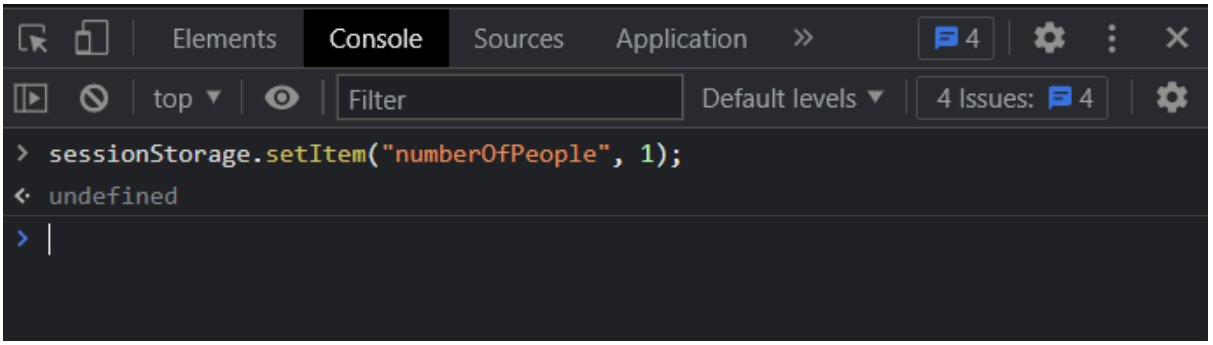
The Web Storage API allows us to store state information in two ways:

1. *sessionStorage* stores state information for each given origin for as long as the browser is open.
2. *localStorage* stores state information for each given origin even when the browser is closed and reopened.

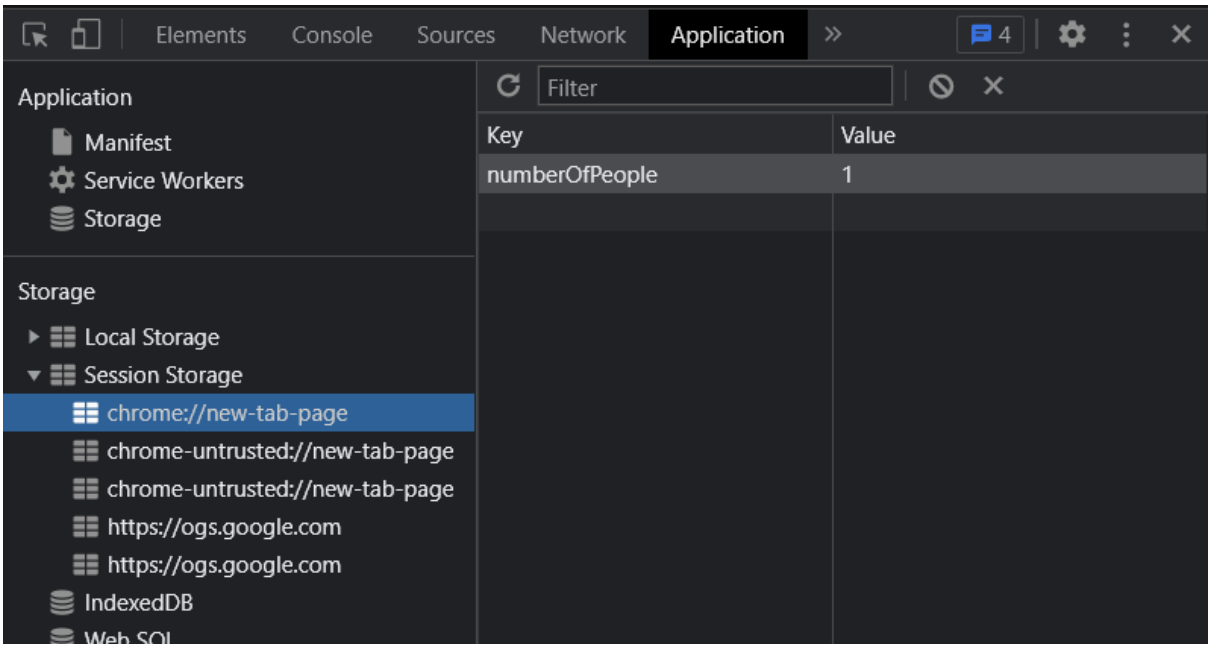
Let's say for example you want to save into session storage the key "numberOfPeople" with a number value of 1. You can add that key-value pair as an item in session storage (saved in the browser) as shown below:

```
sessionStorage.setItem("numberOfPeople", 1);
```

This will add the key value pair {"numberOfPeople", 1} to sessionStorage. Try running the above line of code in the console in your Chrome browser after opening a new Chrome tab (press the F12 key to open the developer tools).



If you enter this line of code as above, you can then navigate to Application → Session Storage → chrome://new-tab-page and you will see the key-value pair saved in the browser.



You could retrieve the value for numberOfPeople from sessionStorage as shown below:

```
let total = parseInt(sessionStorage.getItem("numberOfPeople"));
console.log(total);
```

You will notice that the parseInt() method needs to be used to convert the value to an integer again. This is because items stored using the Web Storage API use JSON and are converted to string data. When retrieving the value it will be returned as a

string. Try running the above code snippet in the console again and you should get the following output:

You can see how we can use the Web Storage API in your web applications to store data in your browser that can be retrieved again. A great example of how this is useful is saving shopping cart items in the browser so that the data can persist whilst you are navigating the website or other websites. Just remember that session storage data is lost after the tab or browser is closed. Local storage data will persist even after the browser has been closed. You would use these 2 based on what your application is trying to achieve.

For more information about how to use sessionStorage see "personObjectEG.js" **here**.

JAVASCRIPT OBJECT NOTATION (JSON)

As stated previously, everything that is transferred over the web is transferred using HTTP. As the name suggests, this protocol can transfer *text*. All data that is transferred across the web is, therefore, transferred as text. We, therefore, cannot transfer JavaScript objects between a web server and a client. *XML* and *JSON* are commonly used to convert JavaScript objects into a format that can be transferred with HTTP.

What is XML?

eXtensible Markup Language (XML) is a markup language that is used to annotate text or add additional information. Tags are used to annotate data. These tags are not shown to the end-user, but are needed by the 'machine' to read and subsequently process the text correctly.

Below is an example of XML. Notice the tags. They are placed on the left and the right of the data you want to markup, so as to wrap around the data. You will notice that these are basically keys and values.

```
<book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
</book>
```

This is the general pattern that we have to follow for all tags in XML.

```
<opening tag>Some text here.</closing tag>
```

Looking at the example of XML above may remind you of HTML. They are both markup languages but, whereas HTML focuses on *displaying* data, XML just *carries data*.

XML files don't do anything except carry information wrapped in tags. We need software to read and display this data in a meaningful way.

XML is used to structure, store, and transport data independent of hardware and/or software.

What is JSON?

JSON, or JavaScript Object Notation, is a syntax for converting objects, arrays, numbers, strings, and booleans into a format that can be transferred between the web server and the client. Like XML, JSON is language independent.

Only text data can be exchanged between a browser and a server. JSON is text and any JavaScript object can be converted into JSON, which can then be sent to the server. The reverse is also true - any JSON data received from the server can also be converted into JavaScript objects. We can therefore easily work with data as JavaScript objects, without any complicated parsing and translations. Data also has to be in a certain format in order to store it. JSON makes it possible to store JavaScript objects as text which is always one of the legal formats.

JSON is much more lightweight than XML. It is shorter, and quicker to read and write. JSON also doesn't use end tags and can use arrays. XML also has to be parsed

(analysed and converted into logical syntactical components in a format such as XML or JSON) with an XML parser, while JSON can be parsed by a standard JavaScript function. (Read more about parsing **here**).

JSON Syntax

The JSON syntax is a subset of the JavaScript syntax. However, this does not mean that JSON cannot be used with other languages. JSON works well with PHP, Perl, Python, Ruby, Java, and Ajax, to name but a few.

Below are some of the JSON Syntax rules:

- Data are in key/value pairs
- Property names must be double-quoted strings
- Data are separated by commas
- Objects are held by curly braces - { }
- Arrays are held by square brackets - [ ]

As mentioned above, JSON data are written as key/value pairs. Each pair consists of a field name or key, which is written in double-quotes, a colon and a value.

Example:

```
"name" : "Jason"
```

The key must be a string, enclosed in double quotes, while the value can be a string, a number, a JSON object, an array, a boolean or null.

JSON uses JavaScript syntax, so very little extra software is needed to work with JSON within JavaScript. The file type for JSON files is ".json" and the MIME type for JSON text is "application/json".

JSON objects

Below is an example of a JSON object:

```
myObj = { "name":"Jason", "age":30, "car":null };
x = myObj.name;
```

The key must be a string, enclosed in double quotes, while the value can be a string, a number, a JSON object, an array, a boolean or null.

Object values can be accessed using the dot (.) notation (as in the example above myObj.name). Object values can also be accessed using square brackets ([ ]) notation (myObj[“name”] in the below example:

Example:

```
myObj = { "name":"Jason", "age":30, "car":null };
x = myObj["name"];
```

JSON uses JavaScript syntax, so very little extra software is needed to work with JSON within JavaScript. The file type for JSON files is **.json** and the MIME type for JSON text is "application/json". (Read more about MIME types **here**).

The dot or bracket notation can be used to modify any value in a JSON object:

```
myObj.age = 31;
```

To delete properties from a JSON object, you use the delete keyword:

```
delete myObj.age;
```

It is also possible to save an array of objects with JSON. See the example below where an array of objects is stored. The first object in the array describes a person object with the name “Tom Smith” and the second object describes a person called “Jack Daniels”: Note that there are even nested objects here:

```
arrayofPersonObjects = [{ "name": { "first": "Tom", "last": "Smith" }, "age":
"21", "gender": "male", "interests": "Programming" },
    { "name": { "first": "Jack", "last": "Daniels" }, "age": "19", "gender":
"male", "interests": "Gaming" }
];
```

If in the above example if we wanted to access the first name of the first object in the array, we could use a mix of the array element location and dot notation as follows:

```
console.log(arrayofPersonObjects[0].name.first)
//Tom would be the expected output here.
```

You can use the *for-in* loop to loop through an object's properties:. Lets say that we have an html paragraph element with the ID “demo” and we want to add append the keys to the paragraph element, we can do the following:

```
myObj = { "name":"Jason", "age":30, "car":null };
// Using the x variable alone will access the key names of each object
propertythe x variable contains the key name as a string for each iteration of
the object. First iteration would be "name", second would be "age" and third
would be "car"
for (x in myObj) {
// innerHTML is used to append each string to the paragraph element with the ID
"demo"
    document.getElementById("demo").innerHTML += x;
}
```

To access thevalues instead of the keys of the object properties in a *for-in* loop, use the bracket notation:

```
myObj = { "name":"Jason", "age":30, "car":null };
for (x in myObj) {
// now the values "Jason", 30 and null are being appended to the "demo"
paragraph element using the innerHTML method
       document.getElementById("demo").innerHTML += myObj[x];
}
```

A JSON object can contain other JSON objects:

```
myObj = {
    "name":"Jason",
    "age":30,
    "cars": {
        "car1":"Ford",
        "car2":"BMW",
        "car3":"VW"
    }
 }
```

To access a value in a nested JSON object, simply use the dot or bracket notation:

```
x = myObj.cars.car2;
//or
x = myObj.cars["car2"];
```

JSON methods

As mentioned before, one of the key reasons for using JSON is to convert certain JavaScript data types (like arrays) into text that can be transferred using HTTP. To be able to do this we use the following two JSON methods:

- `JSON.parse()`

  By parsing the data using **`JSON.parse()`**, the data becomes a JavaScript object. Imagine that you receive the following text from a web server:

```
'{ "name":"Jason", "age":30, "city":"New York"}'
```

  Using the **`JSON.parse()`** function, the text is converted into a JavaScript object:

```
let obj = JSON.parse({ "name":"Jason", "age":30, "city":"New York"}');
```

  If we console.log the obj variable and its type, we can see that the text is converted to an object:

  Consider if we had a paragraph element as follows:

```
<p id="demo"></p>
```

We could use the innerHTML method to then add data from the text converted to an object in our Javascript file like below:

```
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
```

- `JSON.stringify()`

  All data you send to a web server has to be a string. To convert a JavaScript object into a string, you use **`JSON.stringify()`**.

  Imagine that you have the following JavaScript object:

```
let obj = { "name":"Jason", "age":30, "city":"New York"};
```

  Using the **`JSON.stringify()`** function converts this object into a string:

```
let myJSON = JSON.stringify(obj);
```

  `myJSON` is now a string, and can be sent to a server:

```
let obj = { "name":"Jason", "age":30, "city":"New York"};
let myJSON = JSON.stringify(obj);
//myJSON is ready to send via HTTP
```

SPOT CHECK 1

Let's see what you can remember from this section.

1. What is JSON?
2. What are the 5 JSON syntax rules?
3. How do you turn data from a JSON format to an object?
4. How do you turn data from a JSON format to a string?

Instructions

Before you get started please be sure to examine the example files for this task.

Compulsory Task 1

Follow these steps:

- Create a webpage that can be used to let a user store information about a catalogue of music. Consider how you could use the Web storage API to achieve this. The documentation for the Web storage API may also be useful.
    - The user should be able to add information (e.g. artist, title, album, genre etc) about their favourite tracks.
    - All the information about all the tracks added by the user should be listed on the webpage.
    - The user should also be able to remove or edit information for a track.

If you are having any difficulties, please feel free to contact our specialist team **on Discord** for support.

Things to look out for:

1. Make sure that you have installed and set up all programs correctly. You have set up **Dropbox** correctly if you are reading this, but **Visual Studio Code** may not be installed correctly.

Completed the task(s)?

Ask an expert to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?,

Click here to share your thoughts anonymously.

SPOT CHECK 1 ANSWERS

1. JSON, or JavaScript Object Notation, is a syntax for converting objects, arrays, numbers, strings, and booleans into a format that can be transferred between the web server and the client. JSON is language independent.
2.
    a. Data are in key/value pairs
    b. Property names must be double-quoted strings
    c. Data are separated by commas
    d. Objects are held by curly braces - { }
    e. Arrays are held by square brackets - [ ]
3. `JSON.parse()`
4. `JSON.stringify()`