Warsaw University of Technology	Advanced Internet Programming
Project 6 JSON Technology and its Applications	Maciej Iwańczyk 311258
Date: 05.04.2024	Grade:

1. Introduction:

In today's project I had to learn JSON and its application in JavaScript and run and explain the following examples:
https://www.w3schools.com/js/js_json_datatypes.asp
Examples 2,4,6 from
https://opensource.adobe.com/Spry/samples/data_region/JSONDataSetSample.html

2. What is JSON?:

JSON (JavaScript Object Notation) is a lightweight data interchange format inspired by JavaScript object literal syntax. It's commonly used for transmitting data between a server and a web application, but it's not tied to JavaScript specifically; it's a language-independent data format.

3. The code:

3.1 The styles

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JSON Examples</title>
<style>
    body {
        font-family: Arial, sans-serif;
        margin: 0;
        padding: 0;
    }
    header {
        background-color: #333;
        color: #fff;
        padding: 20px;
        text-align: center;
    }
    section {
        margin: 20px;
        padding: 20px;
        border: 1px solid #ccc;
    }
```

```
    h2 {
        margin-top: 0;
    }
    .dataTable {
        width: 100%;
        border-collapse: collapse;
    }
    .dataTable th, .dataTable td {
        border: 1px solid #ddd;
        padding: 8px;
        text-align: left;
    }
    .dataTable th {
        background-color: #f2f2f2;
    }
</style>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Spry/1.6.1/SpryData.js"></script>
</head>
<body>

<header>
    <h1>JSON Examples</h1>
</header>
```

These are just some styles to make it look better. I have also implemented some AJAX scripts into the project.

3.2 The scripts

3.2.1 Example 1

```
<script>
    // Example 1
        const data1 = {"name":"John"};
        const data2 = {"age":30};
        const data3 = {"employee":{"name":"John","age":30,"city":"New York"}};
        const data4 = {"employees":["John","Anna","Peter"]};
        const data5 = {"sale":true};
        const data6 = {"middlename":null};

        // Update HTML with JSON data
        document.getElementById("name").textContent = data1.name;
        document.getElementById("age").textContent = data2.age;
        document.getElementById("city").textContent = data3.employee.city;
        data4.employees.forEach(employee => {
            const li = document.createElement("li");
            li.textContent = employee;
            document.getElementById("employees").appendChild(li);
        });
        document.getElementById("sale").textContent = data5.sale;
        document.getElementById("middlename").textContent = data6.middlename;
```

- data1.name is assigned to the HTML element with the ID "name".
- data2.age is assigned to the HTML element with the ID "age".
- data3.employee.city is assigned to the HTML element with the ID "city".
- data4.employees.forEach() iterates through the array of employee names (data4.employees). For each name, it creates a new <li> element, sets its text content to the employee's name, and appends it to the <ul> element with the ID "employees".
- data5.sale is assigned to the HTML element with the ID "sale".
- data6.middlename is assigned to the HTML element with the ID "middlename".

3.2.2 Example 2

```
// Example 2
var example2 = [
    { "color": "red", "value": "#f00" },
    { "color": "green", "value": "#0f0" },
    { "color": "blue", "value": "#00f" },
    { "color": "cyan", "value": "#0ff" },
    { "color": "magenta", "value": "#f0f" },
    { "color": "yellow", "value": "#ff0" },
    { "color": "black", "value": "#000" }
];

var example2_output = '';
example2.forEach(function(item) {
    example2_output += item.color + '(' + item.value + ') ';
});

$('#example2-output').html('Values from array: ' + example2_output);
```

- example2 is an array of objects. Each object contains two properties: "color" and "value".
- example2_output is initialized as an empty string.
- The .forEach() method is used to iterate over each object in the example2 array. For each object, it concatenates the "color" and "value" properties into the example2_output string.
- After the loop completes, the content of the HTML element with the ID "example2-output" is updated using jQuery's .html() method. The content is set to 'Values from array: ' concatenated with the example2_output string.

3.2.3 Example 4

```
// Example 4
var example4 = {
    "id": "0001",
    "type": "donut",
    "name": "Cake",
    "ppu": 0.55,
    "batters": {
        "batter": [
            { "id": "1001", "type": "Regular" },
            { "id": "1002", "type": "Chocolate" },
            { "id": "1003", "type": "Blueberry" },
            { "id": "1004", "type": "Devil's Food" }
        ]
    },
    "topping": [
        { "id": "5001", "type": "None" },
        { "id": "5002", "type": "Glazed" },
        { "id": "5005", "type": "Sugar" },
        { "id": "5007", "type": "Powdered Sugar" },
        { "id": "5006", "type": "Chocolate with Sprinkles" },
        { "id": "5003", "type": "Chocolate" },
        { "id": "5004", "type": "Maple" }
    ]
};

var example4_output = '<p>Batters:</p><ul>';
example4.batters.batter.forEach(function(item) {
    example4_output += '<li>' + item.type + ' (' + item.id + ')</li>';
});
example4_output += '</ul>';

$('#example4-output').html(example4_output);
```

- The example4 object contains various properties describing a donut, such as its ID, type, name, price per unit (ppu), batters, and toppings.
- example4_output is initialized as a string containing an opening <p> tag followed by the text "Batters:" and an opening <ul> tag.
- The .forEach() method is used to iterate over each object in the batter array, which is nested within the batters property of example4. For each batter object, it appends an <li> element to the example4_output string containing the batter type and ID.
- After the loop completes, a closing tag is appended to the example4_output string.
- Finally, the content of the HTML element with the ID "example4-output" is updated using jQuery's .html() method. The content is set to the example4_output string, which contains the formatted list of batters.

3.2.4 Example 6

```
150     // Example 6
151     var example6 = {
152         "id": "0001",
153         "type": "donut",
154         "name": "Cake",
155         "image": {
156             "url": "images/0001.jpg",
157             "width": 200,
158             "height": 200
159         },
160         "thumbnail": {
161             "url": "images/thumbnails/0001.jpg",
162             "width": 32,
163             "height": 32
164         }
165     };
166
167     var example6_output = '<table class="dataTable">';
168     example6_output += '<tr><th>id</th><th>type</th><th>name</th><th>image.width</th><th>image.height</th><th>image.url</th></tr>';
169     example6_output += '<tr><td>' + example6.id + '</td><td>' + example6.type + '</td><td>' + example6.name + '</td><td>' + example
170     example6_output += '</table>';
171
172     $('#example6-output').html(example6_output);
173 </script>
```

- The example6 object contains information about a donut, including its ID, type, name, image details (URL, width, and height), and thumbnail details (URL, width, and height).
- example6_output is initialized as a string containing the opening <table> tag with a class of "dataTable".
- A header row is added to the table with the <tr> and <th> tags. It includes headers for "id", "type", "name", "image.width", "image.height", and "image.url".
- Another row is added to the table with the <tr> and <td> tags. This row contains the data from the example6 object. Each property value is inserted into a table cell.
- Finally, the closing </table> tag is appended to the example6_output string.
- The content of the HTML element with the ID "example6-output" is updated using jQuery's .html() method. The content is set to the example6_output string, which contains the dynamically generated table.

3.3 The layout

```
<section id="example1">
    <h2>Example 1:</h2>
    <ul>
        <li><strong>Name:</strong> <span id="name"></span></li>
        <li><strong>Age:</strong> <span id="age"></span></li>
        <li><strong>City:</strong> <span id="city"></span></li>
        <li><strong>Employees:</strong>
            <ul id="employees"></ul>
        </li>
        <li><strong>Sale:</strong> <span id="sale"></span></li>
        <li><strong>Middle Name:</strong> <span id="middlename"></span></li>
    </ul>
</section>

<section id="example2">
    <h2>Example 2:</h2>
    <div id="example2-output"></div>
</section>

<section id="example4">
    <h2>Example 4:</h2>
    <div id="example4-output"></div>
</section>

<section id="example6">
    <h2>Example 6:</h2>
    <div id="example6-output"></div>
</section>
```

This is a simple layout for my page presenting the usage of all the examples.

4. The final project:

The webpage: https://zstarwarss.github.io/Project-6-Maciej-Iwanczyk/
The files: https://github.com/ZstarwarsS/Project-6-Maciej-Iwanczyk

I have hosted the page with all the files that can be viewed on GitHub because there are many files so sending them via email would be problematic.