Department of
Computer Science and Engineering

Title: Implementing AJAX and JSON in Web Development

Web Programming Lab

CSE 302

Green University of Bangladesh

1 Objective(s)

- To gather knowledge of JSON.
- Using JSON to send, store, and receive data from the server.
- With the help of Ajax XMLHTTPRequest receiving data from the server on the fly without a page refresh.
- load new JSON data from the server using AJAX technology on-the-fly

2 Problem analysis

A JSON file living on the server has the list of different animal and the extra details about each animal. Create a webpage that has a button, on the click of which it fetches the JSON file from the server and displays the contents of the JSON file, below the button.

Essentially, the contents of the JSON file are loaded on the fly. This means when the page is loaded the first time, there is only an empty div on the page and a button. When the button is clicked, using Javascript, the browser goes to the internet and downloads the information from the server onto the page and creates the HTML. All this happens without a page refresh.

- JSON is used to send, store, and receive data from the server.
- JSON stands for Javascript Object Notation.
- JSON is an array of objects.
- For this Lab exercise, We have to create a JSON file that forked from GitHub at the URL: https://learnwebcode.github.io example/animals-1.json
- Ajax stands for Asynchronous Javascript and XML.
- Ajax is the process of sending and receiving data on the fly without a page refresh.
- The browser uses XMLHTTPRequest to establish a connection to a specified URL and send and receive data.

3 Introduction to JSON and Its Various Methods

3.1 JSON - Evaluates to JavaScript Objects

- The JSON format is almost identical to JavaScript objects.
- In JSON, keys must be strings, written with double quotes:

JSON

```
1 {"name":"John"}
```

- This example is a JSON string:

```
1 '{"name":"John", "age":30, "car":null}'
```

- JSON Array Literals

JSON string:

```
1 '["Ford", "BMW", "Fiat"]'
```

3.2 JSON Methods

- A common use of JSON is to exchange data to/from a web server
- When sending data to a web server, the data has to be a string
- Convert a JavaScript object into a string with JSON.stringify()
- When receiving data from a web server, the data is always a string
- Parse the data with JSON.parse(), and the data becomes a JavaScript object

3.2.1 JSON.stringify()

– Imagine we have this object in JavaScript:

```
1        const obj = {name: "John", age: 30, city: "New York"};
```

– Use the JavaScript function JSON.stringify() to convert it into a string.

```
1        const myJSON = JSON.stringify(obj);
```

Input
string.html

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h2>Create a JSON string from a JavaScript object.</h2>
6  <p id="demo"></p>
7
8  <script>
9  const obj = {name: "John", age: 30, city: "New York"};
10 const myJSON = JSON.stringify(obj);
11 document.getElementById("demo").innerHTML = myJSON;
12 </script>
13
14 </body>
15 </html>
```

Output

```
1  Create a JSON string from a JavaScript object.
2  {"name":"John","age":30,"city":"New York"}
```

3.2.2 JSON.parse()

– Imagine we received this text from a web server:

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

– Use the JavaScript function JSON.parse() to convert text into a JavaScript object:

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

Input
parse.html

```
<!DOCTYPE html>
<html>
<body>

<h2>Creating an Object from a JSON String</h2>

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

<script>
const txt = '{"name":"John", "age":30, "city":"New York"}'
const obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>
</body>
</html>
```

Output

```
Creating an Object from a JSON String
John, 30
```

3.2.3 Array as JSON

When using the JSON.parse() on a JSON derived from an array, the method will return a JavaScript array, instead of a JavaScript object.

Input
array.html

```
<!DOCTYPE html>
<html>
<body>

<h2>Creating an Array from JSON</h2>
<p id="demo"></p>

<script>
const myJSON = '["Ford", "BMW", "Fiat"]';
const myArray = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myArray;
</script>

</body>
</html>
```

Output

```
Creating an Array from JSON
Ford,BMW,Fiat
```

4 Implementation in HTML, JavaScript and CSS

4.1 Step 1: Create the HTML structure.

- Create a button on the web page.

- Create an empty div to load the contents of the JSON file.

```
<button id="btn">Fetch info from three new animals </button>
<div id="animal-info"></div>
```

4.2 Step 2: Create and sent the request.

- Create a new instance of XMLHttpRequest.
- Open a connection and specify if you are sending(POST) or receiving(GET) data using POST/GET and the URL where the JSON data file is located.
- Specify what function or tasks to perform once the connection is LOADED.
- Send the request to the server.

```
var ourRequest = new XMLHttpRequest();
ourRequest.open('GET', 'https://learnwebcode.github.io/json-example/animals-1.
    json');
ourRequest.onload = function(){
console.log(ourRequest.responseText);
};
ourRequest.send();
```

- In Step 2, we logged the output of the JSON to the console, just to test that things are working well so far.
- The XMLHttpRequest property – responseText is used to retrieve the text data received from the server once the request has been received.
- The XMLHttpRequest property – statusText can be used to retrieve a string status message from the server.
- The XMLHttpRequest property – status can be used to retrieve a numerical status code from the server.

4.3 Step 3: Extract the response from the request.

- Store the result of the JSON response into a variable.
- The results are usually one big text string and NOT a JSON object. Using the browser filter JSON.parse, convert the text string into a JSON object.
- Extract the first object from the array.

```
ourRequest.onload = function(){
var ourData= JSON.parse(ourRequest.responseText);
console.log(ourData[0]);
};
```

4.4 Step 4: Hook the button click event.

- Add an event listener to the button's click event so the AJAX is loaded on the button click.

```
var ourRequest = new XMLHttpRequest();
var btn = document.getElementById('btn');
btn.addEventListener('click', function(){
ourRequest.open('GET', 'https://learnwebcode.github.io/json-example/animals-1.
    json');
ourRequest.onload = function(){
var ourData= JSON.parse(ourRequest.responseText);
console.log(ourData[0]);
};
ourRequest.send();
});
```

4.5 Step 5: Build out test HTML where the results are to be displayed.

- Build out the HTML on the page to the empty div created in Step 1- from the JSON objects. As a proof of concept, create a test line in the empty div.
- o keep things organized in the code, the code rendering JS is written in a separate function.

```
var ourRequest = new XMLHttpRequest();
var btn = document.getElementById('btn');
var animalContainar = document.getElementById('animal-info');
btn.addEventListener('click', function(){
ourRequest.open('GET', 'https://learnwebcode.github.io/json-example/animals-1.
    json');
ourRequest.onload = function(){
var ourData= JSON.parse(ourRequest.responseText);
renderAnimal(ourData);
};
ourRequest.send();
});


function renderAnimal(data){
animalContainar.insertAdjacentHTML('beforeend', 'testing');
}
```

4.6 Step 6: Build out the actual HTML where the results are to be displayed.

- Build out the actual HTML on the page-to the empty div created in Step 1- from the JSON objects.
- Use the for-loop to loop through the array and display the objects in the array in a new paragraph tag.

```
function renderAnimal(data){

    var readData = "";

    for (i=0; i < data.length; i++){

        readData += "<p>" + data [i].name +  " is a "  + data[i].species + "
           that likes to eat " ;
    for (ii = 0 ; ii < data [i].foods.likes.length; ii++ ){
```

```
        if (ii== 0) {
            readData += data[i].foods.likes[ii] ;

        }
        else{
            readData += " and " + data[i].foods.likes[ii] ;
        }
    }
    readData += " and dislikes ";

    for (ii = 0 ; ii < data [i].foods.dislikes.length; ii++ ){
        if (ii== 0) {
            readData += data[i].foods.dislikes[ii] ;

        }
        else{
            readData += " and " + data[i].foods.dislikes[ii] ;
        }
    }

    readData +='.</p>';
    }

animalContainar.insertAdjacentHTML ('beforeend', readData);

}
```

4.7 Step 7: Add improvements to the user experience.

- Add counter to make sure that once the button is clicked and the JSON is loaded into the Div, the button is disabled. Otherwise, the table is loaded every time the button gets clicked.
- Add CSS to the button for the enabled/disabled state.

```
var pageCounter = 1 ;
var animalContainar = document.getElementById ("animal-info");
var btn = document.getElementById("btn");

btn.addEventListener("click", function(){
    var ourRequest = new XMLHttpRequest();
    ourRequest.open('GET', 'https://learnwebcode.github.io/json-example/animals-
        '+ pageCounter +'.json');
    ourRequest.onload = function(){
        var ourData = JSON.parse(ourRequest.responseText);
  renderAnimal(ourData);
    };
    ourRequest.send();
    pageCounter++;
    if (pageCounter > 3){
        btn.classList.add("hide-me");
    }

});
```

5 Input

Input of the program is given below.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JASON and AJAX</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <h1 >JASON and AJAX</h1>
        <button id="btn">Fetch info from three new animals </button>
    </header>
    <br>
    <div id="animal-info"></div>
    <script src="ajax.js"></script>

</body>
</html>
```

ajax.js

```
var pageCounter = 1 ;
var animalContainar = document.getElementById ("animal-info");
var btn = document.getElementById("btn");

btn.addEventListener("click", function(){
    var ourRequest = new XMLHttpRequest();
    ourRequest.open('GET', 'https://learnwebcode.github.io/json-example/animals-
       '+ pageCounter +'.json');
    ourRequest.onload = function(){
        var ourData = JSON.parse(ourRequest.responseText);
  renderAnimal(ourData);
    };
    ourRequest.send();
    pageCounter++;
    if (pageCounter > 3){
        btn.classList.add("hide-me");
    }

});

function renderAnimal(data){

    var readData = "";

    for (i=0; i < data.length; i++){

        readData += "<p>" + data [i].name +  " is a "  + data[i].species + "
           that likes to eat " ;
    for (ii = 0 ; ii < data [i].foods.likes.length; ii++ ){
```

```
        if (ii== 0) {
            readData += data[i].foods.likes[ii] ;

        }
        else{
            readData += " and " + data[i].foods.likes[ii] ;
        }
    }
    readData += " and dislikes ";

    for (ii = 0 ; ii < data [i].foods.dislikes.length; ii++ ){
        if (ii== 0) {
            readData += data[i].foods.dislikes[ii] ;

        }
        else{
            readData += " and " + data[i].foods.dislikes[ii] ;
        }
    }

    readData +='.</p>';
    }

animalContainar.insertAdjacentHTML ('beforeend', readData);

}
```

style.css

```
#btn {
color: rgb(17, 16, 16);
background-color: teal;
padding: 10px;
position: relative;
top: 20px;
left: 30px;

}
.hide-me{

    visibility: hidden;
}
```

6 Output

Figure 1: Basic Web Page with a Button

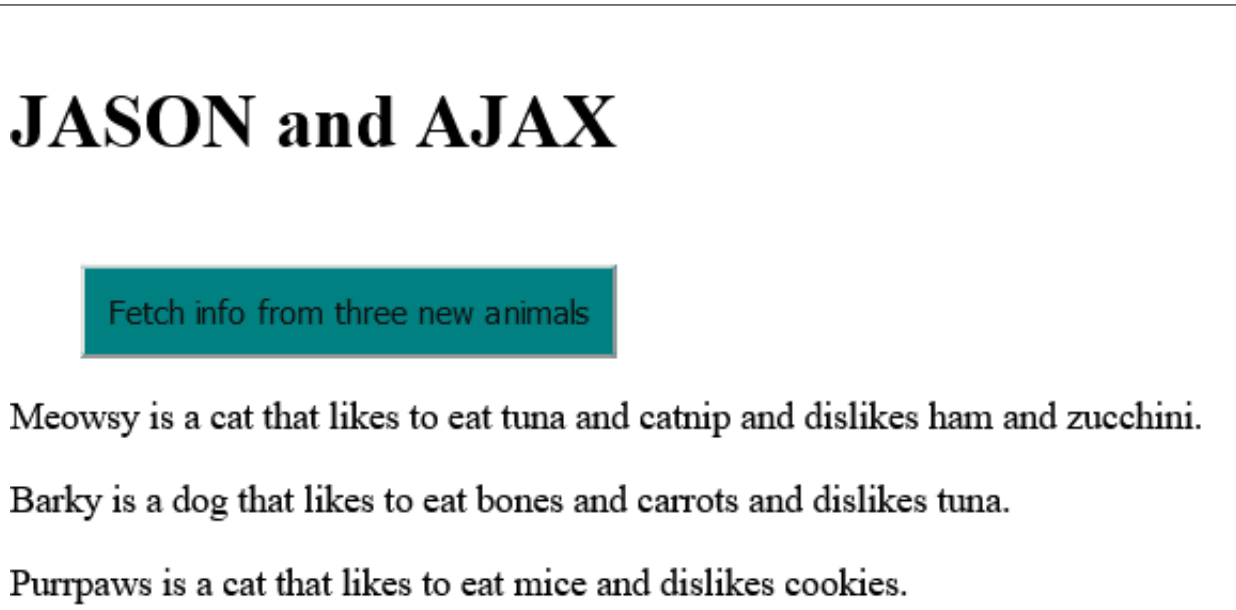


Figure 2: Data Load from Server on First Click

JASON and AJAX

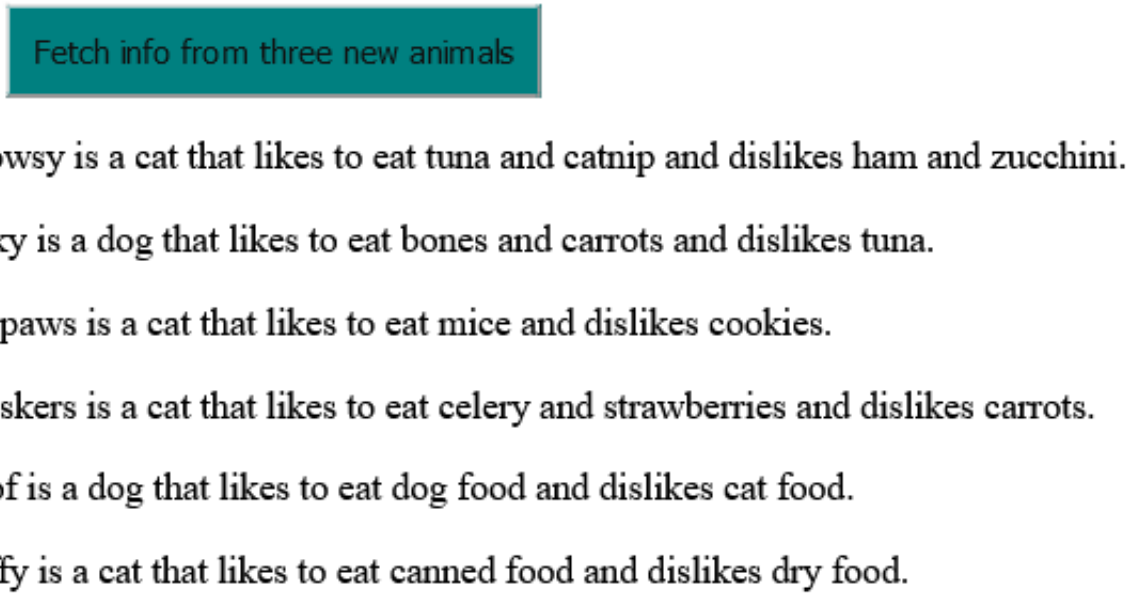


Figure 3: Data Load from Server on Second Click

JASON and AJAX

Meowsy is a cat that likes to eat tuna and catnip and dislikes ham and zucchini.

Barky is a dog that likes to eat bones and carrots and dislikes tuna.

Purrpaws is a cat that likes to eat mice and dislikes cookies.

Whiskers is a cat that likes to eat celery and strawberries and dislikes carrots.

Woof is a dog that likes to eat dog food and dislikes cat food.

Fluffy is a cat that likes to eat canned food and dislikes dry food.

Kitty is a cat that likes to eat fresh food and dislikes stale food.

Pupster is a dog that likes to eat tomatoes and peas and dislikes bread.

Tux is a cat that likes to eat fancy dishes and dislikes basic cat food.

Figure 4: Data Load from Server and Button Disappear

7 Discussion & Conclusion

Based on the objective(s) to understand about JSON and Ajax, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

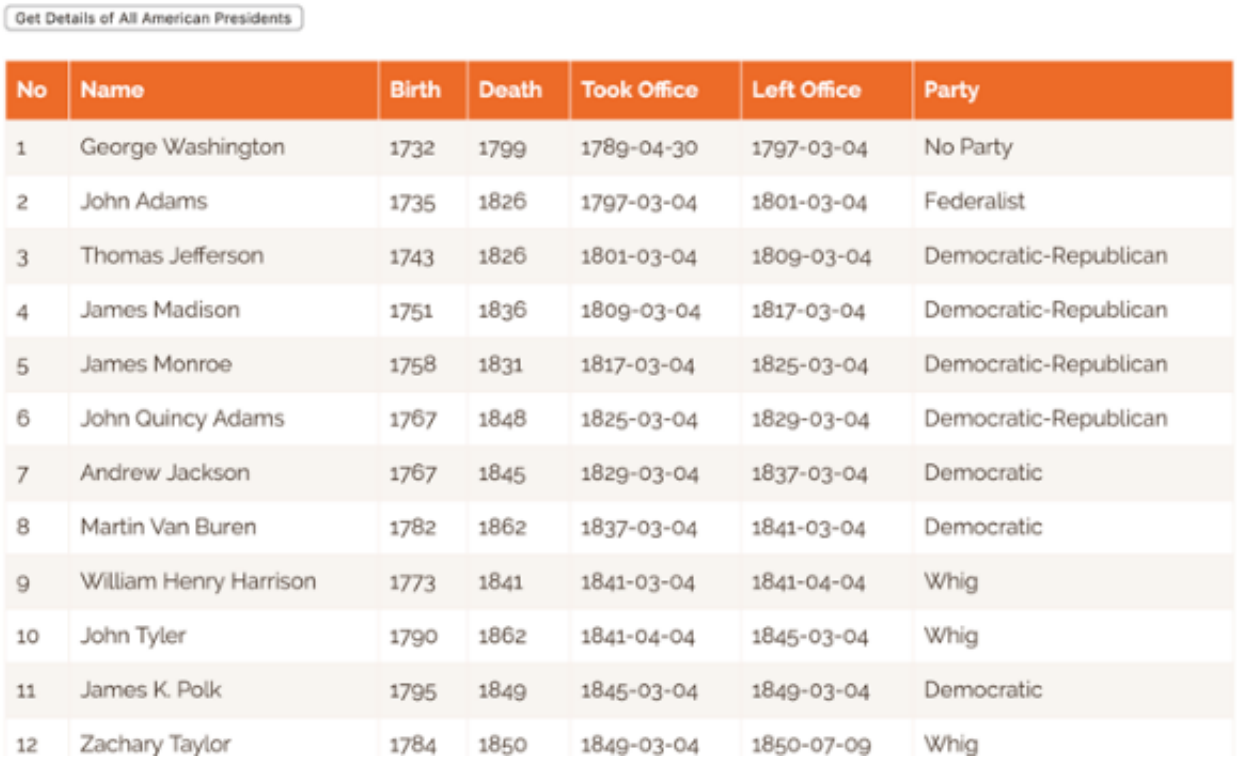
8 Lab Task (Please implement yourself and show the output to the instructor)

1. Develop an interactive web page to load new data on-the-fly from a JSON File (URL: https://github.com/hitch17/sample data/blob/master/presidents.json) in a tabular form Using AJAX and some CSS.

8.1 Problem analysis

A JSON file (URL: https://github.com/hitch17/sample-data/blob/master/presidents.json) living on the server has the list of all the American presidents and the extra details about each president. Create a web page that has a button, on the click of which it fetches the JSON file from the server and displays the contents of the JSON file, below the button in a table.

8.2 Sample output



No	Name	Birth	Death	Took Office	Left Office	Party
1	George Washington	1732	1799	1789-04-30	1797-03-04	No Party
2	John Adams	1735	1826	1797-03-04	1801-03-04	Federalist
3	Thomas Jefferson	1743	1826	1801-03-04	1809-03-04	Democratic-Republican
4	James Madison	1751	1836	1809-03-04	1817-03-04	Democratic-Republican
5	James Monroe	1758	1831	1817-03-04	1825-03-04	Democratic-Republican
6	John Quincy Adams	1767	1848	1825-03-04	1829-03-04	Democratic-Republican
7	Andrew Jackson	1767	1845	1829-03-04	1837-03-04	Democratic
8	Martin Van Buren	1782	1862	1837-03-04	1841-03-04	Democratic
9	William Henry Harrison	1773	1841	1841-03-04	1841-04-04	Whig
10	John Tyler	1790	1862	1841-04-04	1845-03-04	Whig
11	James K. Polk	1795	1849	1845-03-04	1849-03-04	Democratic
12	Zachary Taylor	1784	1850	1849-03-04	1850-07-09	Whig

Figure 5: Load new data in tabular form

9 Lab Exercise (Submit as a report)

- Fetching data from the server.
- Fetching data from the server using XMLHttpRequest and fetch API.

10 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.