Introduction to Getting started with Objects and JSON using JavaScript

Please note this is a fast paced course with prerequisites - it demonstrates how to use and apply JavaScript to connect with various endpoints, make requests and return JSON formatted data demonstrated in interactive fun projects. **Prior coding experience is required** as lessons focus on applying necessary JavaScript functions to create interactions and dynamic content. There will be a focus on DOM based coding and use of JSON and JavaScript Objects.

AJAX allows you to communicate with the server, exchange data and update the page without having to refresh the page.

The two major features of AJAX allow you to do the following:

- Make requests to the server **without reloading the page**
- Receive and work with **data from the server**

Course is loaded with 30 Projects to help you learn and practice connecting to endpoints, returning JSON data, making fetch requests, using JSON data with JavaScript to create Page content, explore GET POST PUT and DELETE (CRUD create, read, update, and delete are the four basic functions of persistent storage) with JavaScript AJAX.

Select, create, manipulate page elements with Modern JavaScript code - make things happen. Create Interactive and fully dynamic web applications driven by JSON data. Get user selections, make the AJAX request from the user select and output to the web page using JavaScript Code.

Explore how to make fun interactive projects to make AJAX requests coming from page data and output the content to the page. Dynamic Page elements generated from AJAX requests to endpoints and return JSON data and use it within JavaScript to update and manipulate DOM page elements. Create Games, Get JSON data and let users explore the data. Dynamically generate page content with JavaScript.

Connect to web endpoints - get JSON data and output that content into your web page with JavaScript

AJAX practice mini projects to help you learn more about JSON data - making fetch requests - debugging and more.

Source Code is included - mini projects are designed to help practice and learn more about AJAX and how you can use it to create content and update web pages.

Section Resources and Code examples

NPM liveServer

https://www.npmjs.com/package/live-server

Open Source Editor IDE

https://code.visualstudio.com/

JavaScript Arrays

Arrays are list-like objects whose prototype has methods to perform traversal and mutation operations. Neither the length of a JavaScript array nor the types of its elements are fixed. Since an array's length can change at any time, and data can be stored at non-contiguous locations in the array,

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

JavaScript Objects

An object is a collection of related data and/or functionality

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics

JSON object

The JSON object contains methods for parsing JavaScript Object Notation (JSON) and converting values to JSON. It can't be called or constructed, and aside from its two method properties, it has no interesting functionality of its own.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

Valid JSON

https://jsonlint.com/

JavaScript Fetch

The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses. It also provides a global fetch() method that provides an easy, logical way to fetch resources asynchronously across the network.

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

JSON parse and JSON stringify

The JSON.parse() method parses a JSON string, constructing the JavaScript value or object described by the string. An optional reviver function can be provided to perform a transformation on the resulting object before it is returned.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse

The JSON.stringify() method converts a JavaScript object or value to a JSON string, optionally replacing values if a replacer function is specified or optionally including only the specified properties if a replacer array is specified.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify

LocalStorage

The read-only localStorage property allows you to access a Storage object for the Document's origin; the stored data is saved across browser sessions. localStorage is similar to sessionStorage, except that while data stored in localStorage has no expiration time, data stored in sessionStorage gets cleared when the page session ends

https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

Setup of the development Environment and LiveServer Localhost setup

How JavaScript Arrays work coding examples of JavaScript Arrays

Lesson Challenge
1. Create an Array
2. Try pop() shift() unshift() push()
3. Clone the array - using
forEach loop and Map
Methods
4. Output results into the console
const arr5 = arr1.map((x)=>x);

Code Example of JavaScript Objects how to use JavaScript Objects

Lesson Challenge

1. Create an Object
2. Add values into the Object
3. Use Bracket Notation as well as Dot notation to return back the values of the object
4. Add an array within the object and return the values of the array.

How to loop through Objects JavaScript Objects Looping and use of values

Lesson Challenge
1. Create an Object
2. Loop through contents of the Object
3. Add an array within the Object as a value
4. Add another object within the Object as a value

JSON and JavaScript Objects methods and code examples JSON Objects in JavaScript

Lesson Challenge
1. Create a Valid JSON file
2. Check JSON at JSON validator
3. Using JavaScript fetch connect to the contents of the file
4. Output JSON file contents to console using JavaScript.

JavaScript JSON methods JSON parse Stringify of String and JavaScript Objects

Lesson Challenge
1. Get fetch content as text()
2. Use JSON parse to turn string object into usable JavaScript Object
3. Use Local storage to save string version of object
4. Get from local storage and turn into usable Object

```
const arr1 = ['one','string',100,false,[]];
arr1[2] = 'test';
//arr1[100] = 100;
//console.log(arr1[2]);
const arr2 = arr1;
//console.log(arr2);
```

```
arr2[4] = 'new value';
//console.log(arr1);
//console.log(arr2);
const arr3 = [];
arr1.forEach((ele,ind)=>{
    console.log(ele);
    arr3[ind] = ele;
})
//console.log(arr3);
arr3[4] = "CHANGED";
//console.log(arr3);
arr1.push('new val');
arr1.unshift('FIRST ITEM');
const val1 = arr1.pop();
const val2 = arr1.shift();
console.log(val1,val2);
const arr4 = arr1.map((el)=>{
    console.log(el);
    return el;
})
console.log(arr4);

const arr5 = arr1.map((x)=>x);
//clone an array
console.log(arr5);



//arr1.length = 0;
//console.log(arr1);

const person = {
    "firstName" : "Laurence",
    "lastName" : "Svekis",
    "x1" : 1,
    "x2" : false,
    "interests" : ["JavaScript","HTML","CSS"],
    "courses" : [{
        "name" : "JavaScript",
        "length" :15
    },{
        "name" : "HTML",
        "length" :10
    },{
        "name" : "CSS",
        "length" :20
    }]
}

const courses = person["courses"];
courses.forEach((course,index)=>{
})
for(const prop in person){
    //console.log(prop);
}
const keys = Object.keys(person);
keys.forEach((key)=>{
    //console.log(person[key]);
})
const vals = Object.values(person);
vals.forEach((val)=>{
    //console.log(val);
})
const entries = Object.entries(person);
for(const arr of entries){
    console.log(arr[0]);
    console.log(arr[1]);
}



/*
console.log(person);
let x = 1;
console.log(person["x"+x]);
x++;
console.log(person["x"+x]);
*/
```

```
{
    "last": "svekis",
    "num": 1,
    "first" : "Laurence"
}

//const person = {"last":"svekis","num":001}
const url = 'j1.json';
let data = '';
myJson();
//console.log(person);

function myJson(){
    fetch(url)
    .then(rep => rep.json())
    .then(json => {
        data = json;
        console.log(data);
    })
}

//const person = {"last":"svekis","num":001}
const url = 'j1.json';
let data = '';

const localData = localStorage.getItem('temp3');

if(!localData){
    myJson();
    console.log('saved to local storage');
}else{
    console.log(localData);
    data = JSON.parse(localData);
    console.log(data);
}

//console.log(person);
```

```
function myJson(){
    fetch(url)
    .then(rep => rep.text())
    .then(json => {
        data = JSON.parse(json);
        //console.log(data);
        let str = JSON.stringify(data);
        //console.log(str);
        localStorage.setItem('temp3',str);
    })
}
```

DOM Manipulation and DOM WebPage updates

JavaScript Coding Review and refresher lessons. Explore the JavaScript code used in the upcoming lessons to select page elements with querySelector, querySelectorAll. Manipulate page content with classList, style, and innerHTML properties. Create interactive content with Click Events by adding event listeners to page elements.

Document Object Model (DOM)

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

The Document Object Model (DOM) connects web pages to scripts or programming languages by representing the structure of a document—such as the HTML representing a web page—in memory. Usually, that means JavaScript, although modeling HTML, SVG, or XML documents as objects are not part of the core JavaScript language, as such.

The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree. With them, you can change the document's structure, style, or content.

Nodes can also have event handlers attached to them. Once an event is triggered, the event handlers get executed.

Window

https://developer.mozilla.org/en-US/docs/Web/API/Window

The Window interface represents a window containing a DOM document; the document property points to the DOM document loaded in that window. A window for a given document can be obtained using the document.defaultView property.

A global variable, window, representing the window in which the script is running, is exposed to JavaScript code.

Document.querySelector()

https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector

The Document method querySelector() returns the first Element within the document that matches the specified selector, or group of selectors. If no matches are found, null is returned.

Array.prototype.forEach()

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach

The forEach() method executes a provided function once for each array element.

```
console.dir(document);
console.log(document.URL);

const hTag = document.querySelector('h1');
console.log(hTag);
hTag.textContent = 'Hello World';
console.log(hTag.textContent);
const divTag = document.querySelector('div');
console.log(divTag);

const divs = document.querySelectorAll('div');
console.log(divs);

divs.forEach(function(ele){
    console.log(ele);
})
divs.forEach(ele => console.log(ele));

divs.forEach((ele,index,arr) => {
    console.log(ele);
    ele.textContent = 'NEW '+ index;
    //console.log(arr);
```

```
})

divs[0].textContent = 'Hello 1';
divs[1].textContent = 'Hello 2';


const divs = document.querySelectorAll('div');
const inValue = document.querySelector('.val');
const hTag = document.querySelector('h1');
const span = document.querySelector('span');

inValue.style.fontSize = '3em';
let counter = 0;
const btn = document.createElement('button');
btn.textContent = 'Click Me';
btn.addEventListener('click',(e)=>{
    const newDiv = document.createElement('div');
    document.body.append(newDiv);
    counter++;

    newDiv.textContent = `${inValue.value} ${counter}` ;
    newDiv.addEventListener('click',myClick);
})

const val1 = span.append(btn);
const val2 = span.appendChild(btn);

console.log(btn);

inValue.addEventListener('click',(e)=>{
    if(inValue.getAttribute('type') =='text'){
        inValue.setAttribute('type','number');
    }else{
        inValue.setAttribute('type','text');
    }

})

divs.forEach((div,ind)=>{
    console.log(div);
    inValue.value = ind;
    div.textContent = `<h2>Hello World</h2> ${inValue.value + 1}`;
    div.innerHTML = `<h2>Hello World</h2> ${inValue.value + 1}`;

    div.addEventListener('click',myClick);

})


function myClick(e){
    console.log(e.target);
    e.target.classList.toggle('box');
}

hTag.addEventListener('click',(e)=>{
    console.log(e.target);
    hTag.style.color = 'white';
    if(hTag.textContent == 'JavaScript'){
        hTag.textContent = 'test';
        hTag.style.backgroundColor = 'blue';
    }else{
        hTag.textContent = 'JavaScript';
        hTag.style.backgroundColor = 'red';
    }
})
```

Project - Create a dynamic list JSON LocalStorage List - project #1

Lesson Challenge
1. Setup JSON file with Data
2. Using fetch Connect to the JSON data
3. Output the JSON data onto your web page
4. Create elements on page dynamically with JavaScript using JSON data

Project -Dynamic list with JSON JavaScript LocalStorage List part 2 - project #1

Lesson Challenge
1. Create global variables to track current list values
2. Get local Storage value of list if it exists otherwise load from JSON file
3. Parse and Stringify as needed to save list contents and JS object list
4. Make list items interactive clickable

Project - Building a Dynamic Interactive JSON List with JavaScript Part 3 - project #1

Lesson Challenge
1. Add option to reload JSON data
2. Create interactions with x to remove list item
3. Add functions to handle list app
4. Add input elements to add to list
5. Add remove and update list values
6. Create dynamic list

Project Final List updates to Local and JSON data list items JavaScript project - project #1

Finalize the list project
Apply styling
Debug and test functionality of the JavaScript list project

Dynamic List Application Loading JSON data Source Code Example

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript List Project</title>
    <style>
        .box {
            padding: 10px;
            width: 300px;
            margin: 5px 0;
            border: 1px solid #ddd;
        }

        .box span {
            width: 20px;
            height: 20px;
            border: 1px solid red;
            padding: 2px;
            border-radius: 10px;
            color: red;
            margin-left: 20px;
        }

        .box span:hover {
            cursor: pointer;
            background-color: cornsilk;
            color: black;
        }

        .confirmed {
            color: green;
        }

        .notConfirmed {
            color: red;
        }
    </style>
</head>

<body>
    <h1>JavaScript</h1>
    <div class="output"></div>
    <script src="app4.js"></script>
</body>

</html>
const output = document.querySelector('.output');
const btn1 = document.createElement('button');
```

```
btn1.textContent = 'Reload JSON';
btn1.addEventListener('click', reloader);
const input1 = document.createElement('input');
const input2 = document.createElement('input');
const btn2 = document.createElement('button');
const div1 = document.createElement('div');
div1.append(input1);
div1.append(input2);
div1.append(btn2);
btn2.textContent = 'Add to List';
input1.setAttribute('placeholder', 'Name');
input2.setAttribute('type', 'number');
input2.value = '1';
document.body.append(div1);
document.body.append(btn1);
btn2.addEventListener('click', addToList);

console.log(output);
const url = 'list.json';
let myList = [];
let localData = localStorage.getItem('myList');
console.log(localData);
window.addEventListener('DOMContentLoaded', () => {
    output.textContent = 'Loading......';
    if (localData) {
        myList = JSON.parse(localStorage.getItem('myList'));
        console.log(myList);
        maker();
    } else {
        reloader();
    }
});

function addToList() {
    console.log(input1.value);
    console.log(input2.value);
    if (input1.value.length > 3) {
        const myObj = {
            "name": input1.value,
            "guests": input2.value,
            "status": false
        }
        const val = myList.length;
        myList.push(myObj);
        savetoStorage();
        makeList(myObj, val);
    }
    input1.value = '';
}


function reloader() {
    fetch(url).then(rep => rep.json())
        .then((data) => {
            myList = data;
            maker();
            savetoStorage();
        })
}


function maker() {
    output.innerHTML = '';
    myList.forEach((el, index) => {
        makeList(el, index);
    });
}

function makeList(item, index) {
    const div = document.createElement('div');
    div.classList.add('box');
    div.innerHTML = `${item.name} #(${item.guests})`;
    output.append(div);
    if (item.status) {
        div.classList.add('confirmed');
    } else {
        div.classList.add('notConfirmed')
    }
    div.addEventListener('click', (e) => {
        div.classList.toggle('confirmed');
        div.classList.toggle('notConfirmed');
        console.log(div.classList.contains('confirmed'));
        if (div.classList.contains('confirmed')) {
            myList[index].status = true;
        } else {
            myList[index].status = false;
        }
        savetoStorage();
    })
    const span = document.createElement('span');
    span.textContent = 'X';
    div.append(span);
    span.addEventListener('click', (e) => {
        console.log(index);
        e.stopPropagation();
        div.remove();
        myList.splice(index, 1);
        savetoStorage();
    })

}

function savetoStorage() {
    console.log(myList);
    localStorage.setItem('myList', JSON.stringify(myList));
}
```

```
[
    {
        "name": "John",
        "guests": 5,
        "status": false
    },
    {
        "name": "Steve",
        "guests": 2,
        "status": false
    },
    {
        "name": "Mary",
        "guests": 1,
        "status": true
    },
    {
        "name": "Mike",
        "guests": 1,
        "status": true
    },
    {
        "name": "Jenny",
        "guests": 5,
        "status": true
    }
]
```

Section Simple JSON endpoints Output to WebPage content with JavaScript

Coding Mini Project create AJAX to Simple JSON file and get JSON with JavaScript - project #2

Practice coding create AJAX request to local JSON file - Create a JSON file valid JSON to JavaScript Object

```
[
  {
    "name": {
      "first": "Laurence",
      "last": "Svekis"
    },
    "age": 40,
    "location": {
      "city": "Toronto",
      "country": "Canada"
    }
  },
  {
    "name": {
      "first": "Lisa",
      "last": "Suekis"
    },
    "age": 30,
    "location": {
      "city": "New York",
      "country": "USA"
    }
  },
  {
    "name": {
      "first": "Johyn",
      "last": "Sekis"
    },
    "age": 50,
    "location": {
      "city": "New York",
      "country": "USA"
    }
  }
]
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript JSON</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Potta+One&display=swap')
;
        body{
            font-family: 'Potta One', cursive;
        }
        .output{
            font-size: 1.2em;
            width:80%;
            margin:auto;
            padding:10px;
        }
    </style>
</head>
<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app1.js"></script>
</body>
</html>
const url = 'https://www.discoveryvip.com/shared/test1.json';
const localUrl = 'people.json';
const btn = document.querySelector('.btn');
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
let attemptCounter = false;
inputVal.style.display = 'none';
btn.textContent = 'Load JSON data';
btn.addEventListener('click',(e)=>{
    getData(url);
})


function getData(urlPath){
    fetch(urlPath).then(rep => {
        return rep.json()
    }).then((json)=>{
        maker(json);
    }).catch(err=>{
        if(!attemptCounter){
            getData(localUrl);
        }
        attemptCounter = true;
        console.log(err);
    })
}

function maker(data){
    output.innerHTML = '<h1>JSON Data</h1>';
    data.forEach((el,index) => {
        console.log(index%2);
        const bg = index%2 == 0 ? '#eee' : '#fff';
        //${JSON.stringify(el)}
        const div = document.createElement('div');
        div.style.backgroundColor = bg;
        div.innerHTML += `<div>${el.name.first}
${el.name.last}</div>`;
        div.innerHTML += `<div>${el.location.city}
${el.location.country}</div>`;
        div.innerHTML += `<div>${el.age} </div>`;
        output.append(div);
    });
}
```

Coding Mini Project AJAX to get JSON data from Wiki API - project #3

Using JavaScript Connect to an API endpoint get Wiki Data and update JavaScript create and update Page content with JSON data.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript JSON</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Potta+One&display=swap')
;
        .box{
            padding:10px;
            margin:auto;
            width:80%;
            border: 1px solid #ddd;
            border-radius: 25px;
        }
        .box > div{
            padding:10px;
            font-size: 1.2em;
        }
    </style>
</head>
<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app2.js"></script>
</body>
</html>
const url =
'https://en.wikipedia.org/w/api.php?action=query&format=json&list=sear
ch&origin=*&srsearch=;';
const btn = document.querySelector('.btn');
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
let attemptCounter = false;
inputVal.value = 'hello';
btn.textContent = 'Load JSON data';
btn.addEventListener('click',(e)=>{
    let searchTerm = inputVal.value || 'JavaScript';
    let tempURL = url + searchTerm;
    console.log(tempURL);
    fetch(tempURL).then((rep)=>{ return rep.json()})
    .then((data)=>{
        console.log(data);
       output.innerHTML = '<div>Results for ' + searchTerm + '</div>';
       output.innerHTML += `Total Results :
${data.query.searchinfo.totalhits}<br>`;
       maker(data.query.search);
    })
})

function maker(data){
    console.log(data);

    data.forEach(el=> {
        console.log(el);
        const div = document.createElement('div');
        div.innerHTML += `<h3><a
href="https://en.wikipedia.org/wiki?curid=${el.pageid}"
target="_blank">${el.title}</a></h3>`;
        div.innerHTML += `<div>Page ID ${el.pageid} | Size ${el.size}
| WordCount ${el.wordcount} </div>`;
        div.classList.add('box');
        div.innerHTML += el.snippet;
        output.append(div);
    });
}
```

Coding Mini Project JSON from WikiMap API examples with JavaScript Code - project #4

Explore how to connect to WikiMap API to get JSON data to update and manipulate page contents using JavaScript. Coding example of practice project to learn more and explore more about connecting to APIs and how to use JavaScript Fetch to get JSON data and output to the web pages.

```
const url =
'http://api.wikimapia.org/?key=example&function=place.getnearest&forma
t=json';
const btn = document.querySelector('.btn');
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
const inputVal2 = document.createElement('input');
inputVal2.setAttribute('type','text');
inputVal2.value = '2.29451'; //lon
document.body.prepend(inputVal2);
const h1 = document.querySelector('h1');
document.body.prepend(h1);
inputVal.value = '48.858252'; //lat
btn.textContent = 'Search Map Lon Lat';
btn.addEventListener('click',(e)=>{
    let lon = inputVal2.value;
    let lat = inputVal.value;
    let tempURL = `${url}&lat=${lat}&lon=${lon}`;
    console.log(tempURL);
    fetch(tempURL).then((res)=>res.json())
    .then((data)=>{
        console.log(data);
        output.innerHTML = '';
        //JSON.stringify(data);
        maker(data.places);
    })
    .catch((err)=>{
        console.log(err);
    })
})

function maker(data){
    data.forEach(el => {
        console.log(el);
        const div = document.createElement('div');
        div.classList.add('box');
        div.innerHTML = `<div>Title
${el.title}<br>${el.urlhtml}</div>`;
        output.append(div);
    });
}
```

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript JSON</title>
    <style>
        .box{
            border: 1px solid black;
            border-radius: 25px;
            padding:20px;
            margin:10px auto;
            width:80%;

        }
        .box:hover{
            opacity: 0.8;
        }
    </style>
</head>
<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app3.js"></script>
</body>
</html>
```

Coding Mini Project Multiple Endpoint Loading JSON data Tester - project #5

Explore connecting to various JSON endpoints - test JSON data to Page content with JavaScript. Using interactive JavaScript web pages to select and make AJAX requests to various endpoints and outputting the JSON data as content to web pages.

```
const btn = document.querySelector('.btn');
const urls = [{
        'url': 'https://www.discoveryvip.com/shared/books2.json',
        'arr': 'books',
        'title': 'Books List'
    },
    {
        'url': 'https://www.discoveryvip.com/shared/people.json',
        'arr': 'people',
        'title': 'Friends List'
    },
    {
        'url': 'https://www.discoveryvip.com/shared/coin.json',
        'arr': 'data',
        'title': 'BitCoin Currency'
    }
];
const h1 = document.querySelector('h1');
```

```
h1.innerHTML = '';
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
inputVal.value = 'test';
btn.textContent = 'Click Me';
btn.addEventListener('click', (e) => {
    //console.log('ready');
    const temp = urls[2];
    //console.log(temp);
    myURL(urls[0]);
})


urls.forEach((ele) => {
    const btn1 = document.createElement('button');
    btn1.classList.add('btn');
    h1.append(btn1);
    btn1.textContent = ele.title;
    btn1.addEventListener('click', (e) => {
        myURL(ele);
    })
})



function myURL(myObj) {
    let url = myObj.url;
    fetch(url)
        .then(rep => rep.text())
        .then((data) => {
            //let val = data.replace(/\s/g,'');
            const json = JSON.parse(data);
            output.innerHTML = url + '<br>';
            maker(json[myObj.arr]);
            //console.log(json);
            //console.log(json);
        })
        .catch((err) => {
            console.log(err);
        })
}


function maker(arr) {
    console.log(arr.length);

    arr.forEach(el => {
        //console.log(el);
        const div = document.createElement('div');
        div.classList.add('box');
        output.append(div);
        const entries = Object.entries(el);
        console.log(entries);
        div.innerHTML = 'Properties : ' + entries.length;
        for (const obj of entries) {
            console.log(obj);
            div.innerHTML += `<br>${obj[0]} : ${obj[1]}`;
        }
    });
}
```

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript JSON</title>
    <style>
        .box{
            padding:10px;
            border: 1px solid #ddd;
            width:80%;
            margin:10px auto;
        }
        .btn{
            padding : 10px;
            font-size: 1em;
            margin:5px;
            border-radius: 15px;
        }
    </style>
</head>
<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app4.js"></script>
</body>
</html>
```

```
{
"data": [
{
"id": 1,
"name": "Bitcoin",
"symbol": "BTC",
"slug": "bitcoin",
"is_active": 1,
"first_historical_data": "2013-04-28T18:47:21.000Z",
"last_historical_data": "2019-04-05T20:44:01.000Z",
"platform": null
},
{
"id": 825,
"name": "Tether",
"symbol": "USDT",
"slug": "tether",
"is_active": 1,
"first_historical_data": "2015-02-25T13:34:26.000Z",
"last_historical_data": "2019-04-05T20:44:01.000Z",
"platform": {
"id": 83,
"name": "Omni",
"symbol": "OMNI",
"slug": "omni",
"token_address": "31"
}
},
{
```

```
"id": 1839,
"name": "Binance Coin",
"symbol": "BNB",
"slug": "binance-coin",
"is_active": 1,
"first_historical_data": "2017-07-25T04:30:05.000Z",
"last_historical_data": "2019-04-05T20:44:02.000Z",
"platform": {
"id": 1027,
"name": "Ethereum",
"symbol": "ETH",
"slug": "ethereum",
"token_address": "0xB8c77482e45F1F44dE1745F52C74426C631bDD52"
}
}
],
"status": {
"timestamp": "2018-06-02T22:51:28.209Z",
"error_code": 0,
"error_message": "",
"elapsed": 10,
"credit_count": 1
}
}
{
"books": [{
"title": "Learn to Code",
"author": "John Smith",
"isbn": "324-23243"
}, {
"title": "The Adventures JSON",
"author": "Jason Jones",
"isbn": "3324-2-444"
}, {
"title": "New Objects",
"author": "Jane Doe",
"isbn": "2343-234-2433"
}]
 }
{
  "people": [
    {
      "first": "Laurence"
      , "last": "Svekis"
    }
    , {
      "first": "Laurence"
      , "last": "Svekis"
    }
    , {
      "first": "Laurence"
      , "last": "Svekis"
    }
    , {
      "first": "Laurence"
      , "last": "Svekis"
    }
  ]
}
```

Coding Mini Project Generate Random User Data Output to Page AJAX and JS Random User API 5 - project #6

Create an interactive Dynamic application that connects to random user API generates pages content depending on user selection. Fully interactive web application using AJAX requests from user page selections - get JSON data and output to yoru web pages with JavaScript Code. Fetch request methods - use of interactive coding to API and customizing the request parameters and request URL from HTML input fields on the webpage.

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript JSON</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Source+Sans+Pro&display=
swap');

        body {
            font-family: 'Source Sans Pro', sans-serif;
        }

        * {
            box-sizing: border-box;
        }

        .box {
            padding: 10px;
            display: inline-block;
            width: 25%;
            text-align: center;
            font-size: 0.7em;

        }

        .box:hover {
            cursor: pointer;
        }

        .box>div {
            border: 1px solid #ddd;
            padding: 10px;
            overflow: hidden;
        }

        .box img {
            width: 100%;
        }
    </style>
</head>

<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app5.js"></script>
</body>

</html>
```

```
const btn = document.querySelector('.btn');
const h1 = document.querySelector('h1');
h1.style.width = '80%';
h1.style.margin = 'auto';
h1.style.textAlign = 'center';
h1.style.border = '5px solid #ddd';
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
//.https://cors-anywhere.herokuapp.com/
const url = 'https://randomuser.me/api/';
inputVal.value = '10';
inputVal.setAttribute('type', 'number');
btn.textContent = 'Click Me';
btn.addEventListener('click', (e) => {
    console.log('ready');
    let val = `?results=${inputVal.value}`;
    adder(url + val);
})

function adder(url) {
    console.log(url);
    fetch(url)
        .then((rep) => {
            return rep.json()
        })
        .then((data => {
            console.log(data);
            output.innerHTML = `<h3>Seed :
${data.info.seed}<br>Results : ${data.info.results}</h3>`;
            maker(data.results);
        }))
}


function maker(data) {
    console.log(data);
    data.forEach(el => {
        console.log(el);
        const loc = el.location;
        const div = eleMaker('div', output, '');
        div.classList.add('box');
        const temp = `${el.name.title} ${el.name.first}
${el.name.last}<br>${el.email}<br>Age : ${el.dob.age}`;
        const temp1 = `<img src="${el.picture.large}">`;
        const temp2 = `${loc.city} ${loc.state} ${loc.country}`;
        div.addEventListener('click', (e) => {
            h1.innerHTML = temp + '<div>' + temp1 + '</div>';
            window.scrollTo({
                top: 0
            });
        })


        eleMaker('div', div, temp);
        eleMaker('div', div, temp1);
        eleMaker('div', div, temp2);
    });
}

function eleMaker(eleTag, parent, contents) {
    const elem = document.createElement(eleTag);
    parent.append(elem);
    elem.innerHTML = contents;
    return elem;
}
```

Coding Mini Project Weather Data App with API JSON data - project #7

Connecting to a weather API to output data with user input values. Create custom query requests to get back JSON data to use on your webpage. AJAX to weather endpoint with page element creation using JavaScript.

```
const btn = document.querySelector('.btn');
const h1 = document.querySelector('h1');
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
const url1 = 'https://samples.openweathermap.org/';
const url2 = 'https://cors-anywhere.herokuapp.com/';
```

```
btn.textContent = 'Click Me';
btn.addEventListener('click', (e) => {
    //console.log('ready');
    getValues(url2 + url1);
})

function getValues(url){
    fetch(url)
    .then(rep => rep.json())
    .then(data => {
        maker(data.products.forecast_5days);
    })
}

function maker(data){
    //console.log(data.docs);
    data.samples.forEach((el)=>{
        getMore(url2 + el);
    })
}

function getMore(url){
    fetch(url)
    .then(rep => rep.json())
    .then(data => {
        adder(data);
    })
}

function adder(data){
    console.log(data.list);
    const city = data.city;
    const div = document.createElement('div');
    output.append(div);
    div.innerHTML = `${city.name} ${city.country} <br>
    ${city.coord.lat} ${city.coord.lon}`;
}
```

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript JSON</title>
    <style>
    </style>
</head>

<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app6.js"></script>
</body>

</html>
```

```
This API enables cross-origin requests to anywhere.

Usage:

/               Shows help
/iscorsneeded   This is the only resource on this host which is served
without CORS headers.
/<url>          Create a request to <url>, and includes CORS headers
in the response.

If the protocol is omitted, it defaults to http (https if port 443 is
specified).

Cookies are disabled and stripped from requests.

Redirects are automatically followed. For debugging purposes, each
followed redirect results
in the addition of a X-CORS-Redirect-n header, where n starts at 1.
These headers are not
```

accessible by the XMLHttpRequest API.
After 5 redirects, redirects are not followed any more. The redirect response is sent back
to the browser, which can choose to follow the redirect (handled automatically by the browser).

The requested URL is available in the X-Request-URL response header.
The final URL, after following all redirects, is available in the X-Final-URL response header.

To prevent the use of the proxy for casual browsing, the API requires either the Origin
or the X-Requested-With header to be set. To avoid unnecessary preflight (OPTIONS) requests,
it's recommended to not manually set these headers in your code.

Demo	:	https://robwu.nl/cors-anywhere.html
Source code	:	https://github.com/Rob--W/cors-anywhere/
Documentation	:	https://github.com/Rob--W/cors-anywhere/#documentation

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any other origins (domain, scheme, or port) than its own from which a browser should permit loading of resources. CORS also relies on a mechanism by which browsers make a “preflight” request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.

An example of a cross-origin request: the front-end JavaScript code served from https://domain-a.com uses XMLHttpRequest to make a request for https://domain-b.com/data.json.

For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts. For example, XMLHttpRequest and the Fetch API follow the same-origin policy. This means that a web application using

```
those APIs can only request resources from the same origin the
application was loaded from unless the response from other origins
includes the right CORS headers.
```

Coding Mini Project Jokes API get fun Jokes for Web Page Content - project #8

Connect to the Chuck Norris open Joke API - get Jokes - Get Category list - return joke data from user selected categories and more. User interactive Dynamic content with JavaScript AJAX requests.

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript JSON</title>
    <style>
        .btns{
            padding:12px;
            margin:5px;
            cursor: pointer;
            font-size: 0.9em;
            border: 1px solid #ddd;
            border-radius: 25px;
            text-transform: capitalize;
        }
        .btns:hover{
            opacity: 0.8;
        }
        input, .btn{
            font-size: 1.5em;
            display: block;
            margin:auto;
            text-align: center;
            width:80%;
        }
        .output{
            padding:20px;
            margin:auto;
            width: 70%;
            font-size: 0.9em;
```

```
                border:1px solid #ddd;
            }

        </style>
    </head>

    <body>
        <h1>JSON</h1>
        <input type="text" class="val">
        <button class="btn">Click</button>
        <div class="output"></div>
        <script src="app7.js"></script>
    </body>

    </html>
    const btn = document.querySelector('.btn');
    const h1 = document.querySelector('h1');
    const output = document.querySelector('.output');
    const inputVal = document.querySelector('.val');
    const url1 = 'https://api.chucknorris.io/jokes/';
    btn.textContent = 'Search';

    buildCats();

    btn.addEventListener('click', (e) => {
        console.log('ready');
        const val1 = inputVal.value || 'test';
        const tempURL = url1 + 'search?query='+val1;
        getJokes(tempURL,val1);
    })

    function buildCats(){
        const urlTemp = url1 + 'categories';
        console.log(urlTemp);
        fetch(urlTemp).then(rep => rep.json())
        .then((data)=>{
            console.log(data);
            h1.innerHTML = '';
            data.forEach((cat)=>{
                const btnTemp = document.createElement('button');
                btnTemp.classList.add('btns');
                btnTemp.textContent = cat;
                h1.append(btnTemp);
                btnTemp.addEventListener('click',(e)=>{

    //https://api.chucknorris.io/jokes/random?category={category}
                    console.log(cat);
                    const tempURL = url1 + 'random?category=' + cat;
                    fetch(tempURL).then(rep=>rep.json())
                    .then((json)=>{
                        output.innerHTML = 'Category : ' + cat + '<hr>';
                        addJoke(json.value);
                    })
                })
            })
        })
    }


    function getJokes(url,searchTerm){
        fetch(url)
        .then(rep => rep.json())
        .then((data)=>{
            output.innerHTML = `${searchTerm} found ${data.total}`;
            console.log(data);
            data.result.forEach((joke) => {
                console.log(joke);
                addJoke(joke.value);
            });
        })

    }


    function getJoke(url){
        fetch(url)
        .then(rep => rep.json())
        .then((data)=>{
            output.innerHTML = '';
            addJoke(data.value);
        })
    }

    function addJoke(val){
        output.innerHTML += val + '<br>';
    }
```

Coding Mini Project Interactive App with Star Wars Data - project #9

Create a fully interactive web application that dynamically loads Star Wars data from an Open API source. Return JSON data and output the data object with JavaScript to your webpage.

```
The for...in statement iterates over all enumerable properties of an
object that are keyed by strings (ignoring ones keyed by Symbols),
including inherited enumerable properties.

const object = { a: 1, b: 2, c: 3 };

for (const property in object) {
  console.log(`${property}: ${object[property]}`);
}

// expected output:
// "a: 1"
// "b: 2"
// "c: 3"


https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Stat
ements/for...in

https://swapi.dev/

<!DOCTYPE html>
<html>

<head>
    <title>JavaScript JSON</title>
    <style>
```

```
* {
    box-sizing: border-box;
}

.btnz {
    padding: 10px;
    border: 1px solid black;
    border-radius: 15px;
    font-size: 0.8em;
    margin: 5px;
    text-transform: uppercase;
    cursor: pointer;
}

.bigText {
    font-size: 1.5em;
    padding: 5px;
    font-weight: bold;
    text-transform: capitalize;
}

.btnz:hover {
    opacity: 0.8;
}

.output {
    text-align: center;
}

.myTitle {
    text-transform: capitalize;
    font-size: 3em;

}

.box {
    padding: 10px;
    text-align: center;
    width: 50%;
    margin: 5px auto;
```

```
            border: 1px solid #ddd;
            cursor: pointer;
            background-color: black;
            color: white;
        }

        .box:hover {
            background-color: red;
        }

        .pages {
            width: 80%;
            text-align: center;
            margin: auto;

        }

        .pages button {
            font-size: 1.2em;
            margin: 5px;
            padding: 10px;
            border-radius: 20px;

        }
    </style>
</head>

<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app8.js"></script>
</body>

</html>
const btn1 = document.querySelector('.btn');
const h1 = document.querySelector('h1');
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
```

```
inputVal.style.display = 'none';
btn1.style.display = 'none';
const mainURL = 'http://swapi.dev/api/';
let endPoint = '';
let endTitle = '';
window.addEventListener('DOMContentLoaded', (e) => {
  //console.log('DOM ready');
  fetch(mainURL).then((rep) => {
    return rep.json();
  }).then((json) => {
    //console.log(json);
    h1.innerHTML = '';
    for (const prop in json) {
      //console.log(`${prop} : ${json[prop]}`);
      const btn = document.createElement('button');
      btn.classList.add('btnz');
      btn.textContent = prop;
      endTitle = prop;
      h1.append(btn);
      btn.urlz = json[prop];
      btn.addEventListener('click', getData);
    }
  })
})

btn1.addEventListener('click', (e) => {
  console.log('ready');
})


function getData(e) {
  //console.log(e.target);
  const el = e.target;
  getJSON(el.urlz);
}


function getJSON(url) {
  endPoint = url;
  fetch(url)
```

```
    .then(rep => rep.json())
    .then(data => {
      console.log(data);
      buildPage(data);
    })
}


function buildPage(data) {
  console.log(data);

  output.innerHTML = `<h1
class="myTitle">${endTitle}</h1><small>${endPoint}</small>`;
  data.results.forEach(el => {
    const div = document.createElement('div');
    div.textContent = el.name || el.title;
    div.classList.add('box');
    div.urlz = el.url;
    div.addEventListener('click', showItem);
    output.append(div);
    console.log(el.name);
  });
  const pages = document.createElement('div');
  pages.classList.add('pages');
  output.append(pages);
  if (data.previous) {
    const btn2 = document.createElement('button');
    btn2.textContent = 'Previous';
    pages.append(btn2);
    btn2.urlz = data.previous;
    btn2.addEventListener('click', (e) => {
      console.log(data.previous);
      getJSON(data.previous);
    });
  }

  const total = Math.ceil(data.count / 10);
  for (let i = 0; i < total; i++) {
    const btn2 = document.createElement('button');
    btn2.textContent = i + 1;
    pages.append(btn2);

    let cleanURL = endPoint.split('?');
    console.log(cleanURL);
    let tempURL = cleanURL[0] + '?page=' + (i + 1);
    btn2.urlz = tempURL;
    btn2.addEventListener('click', (e) => {
      console.log(tempURL);
      getJSON(tempURL);
    });
  }
  if (data.next) {
    const btn2 = document.createElement('button');
    btn2.textContent = 'Next';
    pages.append(btn2);
    btn2.urlz = data.next;
    btn2.addEventListener('click', (e) => {
      console.log(data.next);
      getJSON(data.next);
    });
  }
}

function showItem(e) {
  const el = e.target;
  console.log(el.urlz);
  output.innerHTML = '';
  fetch(el.urlz).then(rep => rep.json())
    .then((data) => {
      //console.log(data);
      for (const prop in data) {
        console.log(`${prop}: ${data[prop]}`);
        let html = (typeof (data[prop]) == 'string') ? data[prop] :
JSON.stringify(data[prop]);
        let propTemp = prop.replace('_', ' ');
        output.innerHTML += `<div><span
class="bigText">${propTemp}</span>: ${html}</div>`;
      }

    })
    .catch((err) => {
      console.log(err);
      output.innerHTML = 'ERROR';
    })
}
```

Coding Mini Project Dynamic Data driven Trivia DataBase Game - project #10

Create a fully functional Trivia game, track game progress, let the user select the number of questions, difficulty and category. Load the data from a JSON endpoint. Output page contents from JSON data. Users can interact and make selections and JavaScript makes the AJAX requests to load requested data for the user. Game functions adding gameplay with scoring and ability to play unlimited rounds of Trivia Questions all dynamically loaded with AJAX.

```
https://opentdb.com/api_config.php

<!DOCTYPE html>
<html>

<head>
    <title>JavaScript JSON</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Work+Sans&display=swap')
;
        *{box-sizing: border-box;}
        body{
            font-family: 'Work Sans', sans-serif;
        }
        .output , .box{
            width:80%;
            margin:auto;
            border:1px solid #ccc;
            background-color: #ddd;
            padding:10px;
        }
        .output > div{
            font-size: 2em;
            margin:auto;

        }
        .box{
            text-align: center;
        }
        .box input  , .box select , .box button{
            font-size: 1.2em;
            border: 1px solid #222;
            line-height: 28px;
            text-transform: capitalize;
            min-height: 35px;
            margin:5px;
        }
        .box input{
            font-size: 1em;
            width:80px;
        }
        h1 {
            text-align: center;
        }
        .output small{
            text-align: center;
            font-size: 0.6em;
        }
        .output button{
            text-align: center;
            padding:10px;
            cursor: pointer;
            border: 1px solid black;
            color:white;
            background-color: #333;
            margin:5px;
            font-size:0.5em;
            border-radius: 10px;

        }

    </style>
</head>
```

```
<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app9.js"></script>
</body>

</html>
const btn1 = document.querySelector('.btn');
const h1 = document.querySelector('h1');
h1.textContent = "Trivia DataBase Game";
const output = document.querySelector('.output');
const output1 = genElement(document.body, 'div', 'Please Make you
Selection<br># Questions');
output1.classList.add('box');
const inputVal = document.querySelector('.val');
output1.append(inputVal);
const sel1 = genElement(output1, 'select', '');
const sel2 = genElement(output1, 'select', '');
output1.append(btn1);
const baseURL = 'https://opentdb.com/api.php?';
const game = {
    que: [],
    question: 0,
    eles: [],
    score:0
};
const cats = [{
    "title": "General",
    "num": 9
}, {
    "title": "Sports",
    "num": 21
}, {
    "title": "Animals",
    "num": 27
}];
const dif = ['easy', 'medium', 'hard'];
```

```
//https://opentdb.com/api.php?amount=10&category=9
window.addEventListener('DOMContentLoaded', (e) => {
    console.log('DOM ready');
    genSelections();
    //testinsert();
    btn1.textContent = 'Start Game';
    inputVal.setAttribute('type', 'number');
    inputVal.value = 10;
})

function genSelections() {
    cats.forEach((cat) => {
        console.log(cat);
        const optEle = genElement(sel1, 'option', cat.title);
        optEle.value = cat.num;
    })
    dif.forEach((d) => {
        console.log(d);
        const optEle = genElement(sel2, 'option', d);
        optEle.value = d;
    })
}

btn1.addEventListener('click', (e) => {
    output1.style.display = 'none';
    h1.textContent = inputVal.value + ' question(s) selected';
    let tempURL =
`${baseURL}amount=${inputVal.value}&difficulty=${sel2.value}&category=
${sel1.value}`;
    console.log(tempURL);
    popPage(tempURL);
})

function testinsert() {
    for (let x = 0; x < 500; x++) {
        let tempArr = [0, 0, 0];
        let ranIndex = Math.floor((Math.random() * (tempArr.length +
1)));
        tempArr.splice(ranIndex, 0, 1);
        output.innerHTML += JSON.stringify(tempArr) + ' : ' + ranIndex
+ '<br>';
    }


}

function popPage(url) {
    fetch(url)
        .then(res => res.json())
        .then(data => {
            game.que = data.results;
            outputPage();
        })
}


function outputPage() {

    console.log(game.question);
    if (game.question >= game.que.length) {
        output.innerHTML = `<div>Your Score was ${game.score} out of
${game.que.length}</div>`;
        game.score = 0;
        output1.style.display = 'block';
        game.question = 0;
    } else {
        output.innerHTML = '';
        let question = game.que[game.question];
        game.question++; //move to next question
        h1.textContent = `Question ${game.question} of
${game.que.length} - SCORE : ${game.score}`;
        console.log(question);
        let answers = question.incorrect_answers;
        let ranIndex = Math.floor(Math.random() * (answers.length +
1));
        console.log(ranIndex);
        answers.splice(ranIndex, 0, question.correct_answer);
        //answers.push(question.correct_answer);
        console.log(answers);
        const mainDiv = genElement(output, 'div', '');
        const que1 = genElement(mainDiv, 'div', question.question);
        game.eles.length = 0;
        const optsDiv = genElement(output, 'div', '');
        answers.forEach(opt => {
            const opt1 = genElement(optsDiv, 'button', opt);
            game.eles.push(opt1);
            if (opt == question.correct_answer) {
                opt1.bgC = 'green';

            } else {
                opt1.bgC = 'red';
            }
            opt1.addEventListener('click', (e) => {
                game.eles.forEach((btnv) => {
                    btnv.disabled = true;
                    btnv.style.backgroundColor = btnv.bgC;
                })
                const message = genElement(optsDiv, 'div', `You got it
Incorrect! <small>${question.correct_answer} was
correct.</small><br>`);
                if (opt == question.correct_answer) {
                    console.log('correct');
                    message.innerHTML = `You Got it Correct!
<small>${opt} was correct.</small><br>`;
                    game.score++;

                    opt1.style.backgroundColor = 'green';
                } else {
                    console.log('wrong');
                    opt1.style.backgroundColor = 'red';
                }
                h1.textContent = `Question ${game.question} of
${game.que.length} - SCORE : ${game.score}`;
                nextQue(message);
                console.log(game);
            })
        });
        /*     game.que.forEach(el => {
                console.log(el);
            }); */
    }
}

function nextQue(parent) {
    const btn2 = genElement(parent, 'button', 'Next Question');
    btn2.addEventListener('click', outputPage);
}

function genElement(parent, eleType, html) {
    const temp = document.createElement(eleType);
    temp.innerHTML = html;
    parent.append(temp);
    return temp;
}
```

Coding Mini Project Stackexchange API tester Request JSON - project #11

Massive API loaded with data and request URL - perfect to practice and get more familiar with complex JSON data - multiple routes for request URLs and loading JSON data to web pages with JavaScript Coding. Explore how you can connect to the Stackexchange API and Request JSON Data to test and build interactive web applications that are data driven and fully dynamic with JavaScript Code. This API has a lot of data and great to familiarize yourself with making AJAX requests with fetch method in JavaScript, Getting JSON data to use within JavaScript Code, Create your own JavaScript code examples of AJAX requests and using JavaScript to create interactive web page content.

```
https://api.stackexchange.com/docs

<!DOCTYPE html>
<html>

<head>
    <title>JavaScript JSON</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Oswald&display=swap');
```

```
* {
    box-sizing: border-box;
}

body {
    font-family: 'Oswald', sans-serif;
}

.tag {
    padding: 5px;
    margin: 5px;
    font-style: italic;
    border: 1px solid #ddd;
    display: inline-block;
}

.tag::before {
    content: "-";
}

.topTitle {
    font-size: 1.2em;
    font-weight: bold;
    margin-bottom: 15px;
    border-bottom: 3px solid black;
    padding-bottom: 10px;
}

.box {
    padding: 15px;
    border: 1px dotted #ddd;
    border-radius: 25px;
    cursor: pointer;
    margin-bottom: 5px;
}

.box:hover {
    background-color: #eee;
}
```

```
        .ans {
            text-align: center;
            font-size: 0.8em;
        }

        .info,
        .output {
            width: 80%;
            border: 1px solid #ddd;
            font-size: 1em;
            margin: auto;
        }

        .output1 {
            width: 80%;
            border: 1px solid #ddd;
            font-size: 1em;
            margin: auto;
            text-align: center;
        }

        input,
        .btn {
            display: block;
            width: 100%;
            font-size: 1.2em;
            height: 45px;
            text-align: center;
        }
    </style>
</head>

<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="app10.js"></script>
</body>
```

```
</html>
const btn1 = document.querySelector('.btn');
const h1 = document.querySelector('h1');
h1.textContent = 'Stackexchange API tester';
const output1 = makeNode(document.body, 'div', '');
output1.classList.add('output1');

const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
inputVal.value = 'javascript';
output1.append(h1);
output1.append(inputVal);
output1.append(btn1);
output1.append(output);

const baseURL = 'https://api.stackexchange.com/';
window.addEventListener('DOMContentLoaded', (e) => {
    //console.log('DOM ready');
    pageLoad();
})

btn1.addEventListener('click', (e) => {
    //console.log('Click ready');
    const searchTerm = inputVal.value;
    const url = baseURL +
`2.2/search?order=desc&sort=activity&intitle=${searchTerm}&site=stacko
verflow`;
    fetch(url)
        .then(res => res.json())
        .then(data => {
            outputItems(data.items);
        })

///2.2/search?order=desc&sort=activity&intitle=javascript&site=stackov
erflow
})

function pageLoad() {
    const url = baseURL +
'2.2/questions?order=desc&sort=activity&site=stackoverflow';
```

```
    //console.log(url);
    fetch(url)
        .then(rep => rep.json())
        .then((data) => {
            outputItems(data.items);
        })
        .catch((err) => {
            //console.log(err);
        })
}

function outputItems(data) {
    ////console.log(data);
    output.innerHTML = '';
    data.forEach(item => {
        outputPage(item);
    });
}

function outputPage(data) {
    //console.log(data);
    const main = makeNode(output, 'div', '');
    main.classList.add('box');
    const ele = makeNode(main, 'div', data.title);
    ele.classList.add('topTitle');
    const ansCount = makeNode(main, 'div', `Answers
${data.answer_count}`);
    ansCount.classList.add('ans')
    ele.qid = data.question_id;
    const quesID = makeNode(main, 'div', `QID ${ele.qid}`);
    if (data.question_id) {
        main.addEventListener('click', (e) => {
            getByID(data.question_id);
        });
    } else {
        main.style.backgroundColor = '#ddd';
    }
    data.tags.forEach((tag) => {
        const span = makeNode(main, 'span', tag);
        span.classList.add('tag');
    })
    //console.log(ele);
}

function makeNode(parent, typeEle, html) {
    const element = document.createElement(typeEle);
    element.innerHTML = html;
    return parent.appendChild(element);
}

function getByID(qid) {


    if (qid) {
        const url1 = baseURL + '2.2/questions/' + qid +
'?order=desc&sort=activity&site=stackoverflow';
        const url2 = baseURL + '2.2/questions/' + qid +
'/answers?order=desc&sort=activity&site=stackoverflow';
        let itemInfo = {};
        fetch(url1)
            .then(rep => rep.json())
            .then((data) => {
                itemInfo = data;
                return fetch(url2)
            })
            .then(res => res.json())
            .then((data) => {
                console.log(itemInfo.items[0]);
                console.log(data.items);
                buildPageData(itemInfo.items[0], data.items);
            })
            .catch((err) => {
                //console.log(err);
            })

////2.2/questions/55986738/answers?order=desc&sort=activity&site=stack
overflow
    } else {
        console.log('No ID');
    }
}

function buildPageData(que, ans) {
    console.log(que);
    console.log(ans);
    output.innerHTML = '';
    const title = makeNode(output, 'div', `<h2>${que.title}</h2>`);
    const qid = makeNode(output, 'div', `<div>Question ID :
${que.question_id}</div>`);
    const link = makeNode(output, 'a', `${que.link}`);
    link.setAttribute('href', que.link);
    link.setAttribute('target', '_blank');
    const total = makeNode(output, 'div', `<div>Answers  :
${que.answer_count}</div>`);
    const answerDiv = makeNode(output, 'div', '');
    answerDiv.classList.add('info');
    ans.forEach((answer, index) => {
        console.log(answer);
        const rating = answer.owner.accept_rate || '-';
        const html = `
        <hr>
        Answer # ${index+1}<br>
        Answer ID ${answer.answer_id}<br>
        Owner : ${answer.owner.display_name} (${rating})
        `;
        const div1 = makeNode(answerDiv, 'div', html);
    });
}
```

Coding Mini Project Select Country Data from API   - project #12

Connect to a countries API that lists over 250 countries with a complex JSON object rich with data including objects and image paths and arrays. Chunk page data into multiple pages allowing the user to select and interact with JSON data content.

Lesson Tasks

1. Connect to API return JSON data
2. Make request from User Button output JSON data onto web page
3. Dynamically Load content into separate pages
4. Allow user to navigate within pages of content

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript JSON</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Noto+Sans+JP&display=swa
p');
        * {
            box-sizing: border-box;
        }
        body{
            font-family: 'Noto Sans JP', sans-serif;
        }
        .box{
            width:80%;
            text-align: center;
            padding: 10px;
            margin:10px auto;
        }
        .box img{
            max-width: 100%;
            width:200px;
        }
        .box h2{
            text-transform: uppercase;
        }
        .pgs{
            padding:10px;
            border: 1px solid #ddd;
            border-radius: 10px;
            cursor: pointer;
            display: inline-block;
            width:50px;
            color:white;
            background-color: black;
            text-align: center;
        }
    </style>
</head>
```

```
<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="apps.js"></script>
</body>

</html>
const btn = document.querySelector('.btn');
const h1 = document.querySelector('h1');
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
const page = {json:{},page:1,per:10,arr:[]};
const baseurl = 'https://restcountries.eu/rest/v2/';
btn.textContent = 'Load Pages';
inputVal.style.display = 'none';
h1.textContent = 'Load Country Info';
btn.addEventListener('click', (e) => {
    console.log('ready');
    const para = 'all';
    const url = baseurl + para;
    fetch(url)
        .then(rep => rep.json())
        .then(data => {
            createPages(data);
        })
})

function createPages(data){
    page.arr.length = 0;
    //let ptotal = data.length / page.per;
    //console.log(ptotal);
    for(let i=0;i<data.length;i+=page.per){
        let tempArr = data.slice(i,i+page.per);
        //console.log(tempArr);
        page.arr.push(tempArr);
    }
    //console.log(page);
    loadPages();
```

```
}

function loadPagination(){
    const main = createNode(output,'div','');
    for(let i=0;i<page.arr.length;i++){
        const pg = createNode(main,'div',i+1);
        pg.classList.add('pgs');
        if(page.page == i+1){
            pg.style.backgroundColor = 'red';
        }
        pg.addEventListener('click',(e)=>{
            console.log(i+1);
            page.page = i+1;
            loadPages();
        })
    }
}


function loadPages(){
    output.innerHTML = '';
    console.log(page.arr,page.page);
    page.arr[page.page-1].forEach(el => {
        pageEl(el);
    });
    loadPagination();
}

function pageEl(data){
    console.log(data);
    const main = createNode(output,'div','');
    main.classList.add('box');
    const title = createNode(main,'div',`<h2>${data.name}</h2>`);
    title.style.color = 'red';
    const flag = createNode(main,'img','');
    flag.setAttribute('src',data.flag);
    let html1 = `<div>Population : ${data.population}</div>`;
    html1 += `<div>Currency : ${data.currencies[0].name}
${data.currencies[0].symbol}</div>`;
```

```
    const stats = createNode(main,'div',html1);
}


function createNode(parent,elType,html){
    const ele = document.createElement(elType);
    parent.append(ele);
    ele.innerHTML = html;
    return ele;
}
```

Coding Mini Project User Search Country by Name get JSON data - project #13

Dynamic and interactive page content with JavaScript - loading JSON data into JavaScript coding objects and outputting and creating page element content with JavaScript DOM.

Lesson Tasks

1. Connect to API return JSON data
2. Customize request parameters according to user input value
3. Load all object details per item on user selection
4. Page object data from one function to another.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript JSON</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Noto+Sans+JP&display=swa
p');
        * {
            box-sizing: border-box;
        }
        body{
            font-family: 'Noto Sans JP', sans-serif;
        }
        .box{
            width:80%;
            text-align: center;
            padding: 10px;
            margin:10px auto;
            cursor: pointer;
        }
        .box img{
            max-width: 100%;
            width:200px;
        }
        .box:hover{
            background-color: blanchedalmond;
        }
        .box h2{
            text-transform: uppercase;
        }
        .info {
            width:80%;
            text-align: center;
            padding: 10px;
            font-size: 1.2em;
            margin:10px auto;
            border: 1px solid #ddd;
            border-radius: 10px;
        }

        .pgs{
            padding:10px;
            border: 1px solid #ddd;
            border-radius: 10px;
            cursor: pointer;
            display: inline-block;
            width:50px;
            color:white;
            background-color: black;
            text-align: center;
        }
    </style>
</head>
<body>
    <h1>JSON</h1>
    <input type="text" class="val">
    <button class="btn">Click</button>
    <div class="output"></div>
    <script src="apps2.js"></script>
</body>

</html>
const btn = document.querySelector('.btn');
const h1 = document.querySelector('h1');
const output = document.querySelector('.output');
const inputVal = document.querySelector('.val');
const page = {json:{},page:1,per:10,arr:[]};
const baseurl = 'https://restcountries.eu/rest/v2/';
btn.textContent = 'Search by Name';
h1.textContent = 'Search Country Info';
inputVal.value = 'united';
btn.addEventListener('click', (e) => {
    console.log('ready');
    const para = 'name/'+inputVal.value;
    const url = baseurl + para;
    fetch(url)
        .then(rep => rep.json())
        .then(data => {
            loadPages(data);
        })
})

function makeaPage(data){
    output.innerHTML = '';
    const main = createNode(output,'div','');
    main.classList.add('info');
    objOutput(data,main);
}


function objOutput(obj,parent){
    Object.keys(obj).forEach(key => {
        console.log(key); //object key
        console.log(obj[key]); //value
        console.log(typeof(obj[key])); //data type
        let val = obj[key];
        if(typeof(val) == 'object'){
            val = JSON.stringify(val);
        }
        createNode(parent,'div',val);
    })

}


function loadPages(data){
    output.innerHTML = '';
    console.log(data);

    data.forEach(el => {
        pageEl(el);
    });
}

function pageEl(data){
    console.log(data);
    const main = createNode(output,'div','');
    main.classList.add('box');
    main.addEventListener('click',(e)=>{
        makeaPage(data);
    })
    const title = createNode(main,'div',`<h2>${data.name}</h2>`);
    title.style.color = 'red';
    const title2 = createNode(main,'div',`${data.nativeName}`);
    createNode(main,'div',`${data.subregion}`);

    const flag = createNode(main,'img','');
    flag.setAttribute('src',data.flag);
    let html1 = `<div>Population : ${data.population}</div>`;
    html1 += `<div>Currency : ${data.currencies[0].name}
${data.currencies[0].symbol}</div>`;
    const stats = createNode(main,'div',html1);
}


function createNode(parent,elType,html){
    const ele = document.createElement(elType);
    parent.append(ele);
    ele.innerHTML = html;
    return ele;
}
```

JSON AJAX JavaScript QUIZ application

Explore how to build a dynamic quiz application that gets JSON data and builds a fully interactive quiz

Introduction to JavaScript Quiz Game Project - project #14

Create a fully interactive quiz game using JavaScript with JSON data from server endpoint. Connect to an API - return back response data to generate the quiz game using the values from the data. Build interactive content track progress randomize question options. Use JavaScript DOM manipulation to select, create and update page elements. Check correct answers, score the player, build the gameplay loading dynamically from the JSON data. Load the content from a Google Sheet - create the full gameplay which can customize the questions and create the JavaScript game progress all using the JSON data. Make AJAX request with JavaScript fetch to the Google Sheet data endpoint , return JSON data and use with JavaScript to create a Quiz game.

How to use JavaScript JSON data to build and setup a Quiz

Lesson Tasks
1. Create a Quiz JSON File
2. Setup HTML Page
3. Add Window Event listener
4. Make Fetch Request to get JSON file Data
5. Load JSON data into the console.

How to Create Interactive Content from JSON with JavaScript DOM manipulation

Lesson Tasks
1. Loop through the data
2. Get object details
3. rebuild into new array with objects and boolean for correct options
4. Check New object in console and within JSON linter
5. Review new structure.

How to add Page Elements with JavaScript from JSON data

Lesson Tasks
1. Add Button to start Quiz
2. Create event listener for button
3. Add question and options
4. Make options clickable and check for the correct answer.

How to Randomize the Order of Array Items with JavaScript

Lesson Tasks
1. Randomize the order of the options for the question
2. Apply styling to the questions and options
3. Debug and try the quiz to ensure working as expected

Creating GamePlay Loading Next Question in Data with JavaScript

Lesson Tasks
1. Apply Styling to selected options
2. Add and Remove event listeners for options
3. Provide option to user to move to next question
4. Color the options for feedback to the user on the selection

Debugging and Fixing Code adding Scoring to Game

Lesson Tasks
1. Debug application
2. Adding Scoring values
3. Fix code blocks
4. Play game to ensure functionality
5. Test and Test again.

How to Review and Finalize JavaScript Game Project

Final review and updates with tweaks for application gameplay.

How to Load JSON source data from a Google Sheet File

Lesson Tasks
1. Create a Google Sheet
2. File > Publish to the Web > Sheet > HIT publish
3. Get ID from your sheet
4. Output as JSON sheet data

- **Bonus PDF Guide how to create JSON output endpoint from Google Sheets**

Getting Google Sheet Contents as JSON to JavaScript Object

Lesson Tasks
1. Load JSON from sheet
2. Extract values needed from the sheet data - rows and columns of data
3. Check the object key values and retrieve property values
4. Rebuild the JavaScript Object for application.

How to create a Quiz game using Sheet Data in JavaScript Quiz

Lesson Tasks
1. Update sheet data
2. add Styling as needed, use Javascript from Quiz app earlier lessons
3. Try and debug as needed.

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript Quiz Project</title>
    <style>
        .box {
            display: inline-block;
            padding: 5px;
            border: 1px solid #ddd;
            border-radius: 10px;

            margin: 5px;
            min-width: 100px;
            text-align: center;
        }
        .boxCursor {
            cursor: pointer;
        }

        .boxCursor:hover {
            background-color: azure;
        }

        .que {
            font-size: 1.8em;
            margin-bottom: 15px;
        }

        .btn {
            display: block;
            width: 50%;
            margin: auto;
            margin-top: 20px;
            border-top: 1px solid black;
            padding: 10px;
        }
    </style>
</head>

<body>
    <h1>JavaScript Quiz</h1>
    <div class="output"></div>
    <button class="btn">Start Game</button>
    <script src="app7.js"></script>
</body>

</html>
const id = '1H2xq8J5uZKDABgu1JlvV-nVWKqZoV8_UpB_D_snOrLo';
const url =
'https://spreadsheets.google.com/feeds/list/'+id+'/1/public/values?
alt=json';
const questions = [];
const output = document.querySelector('.output');
const btn = document.querySelector('.btn');
let cur = 0;
const player = {
    score: 0,
    answers: []
}
const holder = [];
const totalOutput = document.querySelector('h1');

btn.addEventListener('click', (e) => {
    if (cur >= questions.length) {
        let html = `<hr><h1>Score = ${player.score}</h1>`;
        player.answers.forEach((el) => {
            let bg = el.correct ? 'green' : 'red';
            html += `<div style="background:${bg}">Question :
${capitalizeText(el.question)}? <br>`;
            html += `Response : ${el.response}
(${el.correctAnswer})<br>`;
            html += `Result : ${el.correct} </div><br>`;
        })
        output.innerHTML = html;
    } else {
        newQuestion();
    }
    btn.style.display = 'none';
})

window.addEventListener('DOMContentLoaded', () => {
    //console.log('ready');
    loadQuestions();
})

function capitalizeText(str) {
    return str.charAt(0).toUpperCase() + str.slice(1);
}

function newQuestion() {
    updateScore();
    const el = questions[cur];
    el.options.sort(() => {
        return 0.5 - Math.random()
    });
    console.log(cur);
    console.log(questions.length);
    console.log(questions[cur]);
    output.innerHTML = '';
    const que1 = document.createElement('div');
    que1.classList.add('que');
    let strOutput = capitalizeText(el.question);
    console.log(strOutput);

    const ans1 = document.createElement('div');
    que1.textContent = strOutput + '?';
    holder.length = 0;
    el.options.forEach((ans) => {
        const div = document.createElement('div');
        holder.push(div);
        div.correctAnswer = el.correct;
        div.textContent = ans.response;
        div.classList.add('box');
        div.classList.add('boxCursor');
        div.correct = ans.correct;
        div.addEventListener('click', selOption);
        ans1.append(div);
    })
    output.append(que1);
    output.append(ans1);
}

function selOption(e) {
    //track the progress
    console.log(e);
    const tempObj = {
        "question": questions[cur].question,
        "response": e.target.textContent,
        "correctAnswer": e.target.correctAnswer
    }
    endTurn();
    if (e.target.correct) {
        player.score++;
        updateScore();
        tempObj.correct = true;
        e.target.style.backgroundColor = 'green';
    } else {
        e.target.style.backgroundColor = 'red';
        tempObj.correct = false;
    }
    player.answers.push(tempObj);
    e.target.style.color = 'white';
    nextBtn();
    console.log(player);
}

function updateScore() {
    totalOutput.innerHTML = `${cur+1} out of ${questions.length}
Score: ${player.score}`;
}

function endTurn() {
    holder.forEach((el) => {
        el.removeEventListener('click', selOption);
        el.style.backgroundColor = '#ddd';
        el.classList.remove('boxCursor');
    })
}

function nextBtn() {
    btn.style.display = 'block';
    cur++;
    if (cur >= questions.length) {
        btn.textContent = 'See Score';
    } else {
        btn.textContent = 'Next Question';
    }
}

function loadQuestions() {
    fetch(url).then(rep => rep.json())
        .then((data) => {
            //console.log(data.feed.entry);
            data.feed.entry.forEach(el => {
                //console.log(el.content['$t']);
                let mainTemp = {options:[]};
                for(let key in el){
                    if(key.substring(0,3) == 'gsx'){
                        let header = key.slice(4);
                        let val = el[key]['$t'];
                        if(header=='question'){
                            mainTemp.question = val;
                        }else if(header=='correct'){
                            mainTemp.correct = val;
                            let temp = {
                                "response": val,
                                "correct": true
                            };
                            mainTemp.options.push(temp);
                        }else{
                            if(val.length>0){
                                let temp = {
                                    "response": val,
                                    "correct": false
                                };
                                mainTemp.options.push(temp);
                            }
                        }
                        //console.log(header);
                        //console.log(el[key]['$t']);
                    }
                }
                questions.push(mainTemp);
            });
            console.log(questions);
            //document.write(JSON.stringify(questions));
        })
}
```

Final Game Play How to Download text Report on Game Score

Lesson Tasks
1. Add option to download report at the end of quiz
2. Generate the report from player data array
3. Output the report as a csv file that downloads to the player computer

Web APIs Practice Mini Applications with JavaScript JSON data to Web Pages

Coding Mini Project Creating a Joke Generating Web page from JSON data - project #15

Explore how to connect to an api with limited requests per hour. Build the app in less than 10 requests. Get JSON data - output as JavaScript object create page content from data.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript API</title>
    <style></style>
</head>
<body>
    <div class="container">
        <input type="text" class="val">
        <button class="btn">Click</button>
        <div class="output"></div>
    </div>
    <script src="code1.js"></script>
</body>
</html>
const btn = document.querySelector('.btn');
const inpEle = document.querySelector('input');
inpEle.style.display = 'none';
const output = document.querySelector('.output');
btn.textContent = 'Get the Joke of the Day';
const url = 'https://api.jokes.one/jod';
```

```
btn.addEventListener('click',(e)=>{
    fetch(url).then(res => res.json())
    .then(data =>{
        console.log(data.contents);
        if(data.success.total > 0){
            outputJoke(data.contents.jokes);
        }
        console.log(data);
    }).catch((err)=>{
        console.log(err);
    })
})
function outputJoke(data){
    console.log(data);
    const joke = data[0];
    output.innerHTML = `<h1>${joke.description}</h1>`;
    output.innerHTML += `<div>${joke.joke.title}</div>`;
    output.innerHTML += `<div>${joke.joke.text}</div>`;
}
```

Coding Mini Project multiple endpoints for JSON data user selected content. - project #16

Using an array data of items, connect to different endpoints, allowing the user to select the content they want to load. Dynamically create a user interaction list of urls and allow user to interact and output the selected content.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Project</title>
    <style>

    </style>
</head>
<body>
    <h1>JavaScript</h1>
    <div class="output"></div>
    <button class="btn">Click Me</button>
    <script src="pro1.js"></script>
```

```
</body>

</html>

const output = document.querySelector('.output');
const btn1 = document.querySelector('.btn');
const urls = ['test.txt','test1.txt','test2.txt'];
const headerOne = document.querySelector('h1');
const sel = document.createElement('select');
output.append(sel);
urls.forEach((el)=>{
    const opt = document.createElement('option');
    sel.append(opt);
    opt.value = el;
    opt.textContent = el;
})

btn1.addEventListener('click',(e)=>{
    getFile(sel.value);
})

function getFile(url){
    fetch(url).then(rep => rep.text())
    .then(data =>{
        headerOne.textContent = data;
    })
}
```

Coding Mini Project connecting to the Github API loading JSON data - project #17

Create a dynamic interactive web user interface connecting to Github data and allowing the user to load selected content. Page content is interactive and content is driven from JSON data provided by the GitHub API.

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript Project</title>
```

```
</head>

<body>
    <h1>JavaScript</h1>
    <div class="output"></div>
    <input class="searchTerm">
    <button class="btn">Click</button>


    <script src="pro2.js"></script>
</body>

</html>
const output = document.querySelector('.output');
const btn1 = document.querySelector('.btn');
//const url = 'https://api.github.com/repos/twbs/bootstrap';
//const url = 'pro2.json';
const url = 'https://api.github.com/search/repositories';
const searchTerm = document.querySelector('.searchTerm');

window.addEventListener('DOMContentLoaded',(e)=>{
    console.log('ready');
    fetch('https://api.github.com/zen').
    then(rep => rep.text()).
    then(message => {
        document.querySelector('h1').textContent = message;
    })
})

btn1.addEventListener('click',(e)=>{
    const val = searchTerm.value;
    const queryString = url + '?q='+encodeURIComponent(val);
    console.log(queryString);
    fetch(queryString)
    .then((rep)=>rep.json())
    .then((data)=>{
        outputArray2(data);
    })
    .catch((error)=>{
        console.log('Fetch problem : '+error.message);
    })
})

function outputArray2(data){
    console.log(data.items);
    if(data.items.length>0){
        data.items.forEach(element => {
            outputContenttoPage(element);
            output.innerHTML +=  '<hr>';
        });
    }
}

function outputArray(data){
    console.log(data);
    data.forEach(element => {
        outputContenttoPage(element);
        output.innerHTML +=  '<hr>';
    });
}

function outputContenttoPage(data){
    console.log(data);
    let html = `
        ${data['name']} <br>
        ${data['id']} <br>
        ${data['owner']['id']}<br>
        <a href="${data['html_url']}"
target="_blank">${data['html_url']}</a><br>
        `;
    output.innerHTML +=  html;
}
```

Coding Mini Project Generate interactive Game with Dynamic Wordlist - project #18

Loading game data dynamically from a Google Sheet. How to share Google Sheet data for web applications.

Coding Mini Project Creating a Game with Sheets Data - project #18

How to create interactive content and gameplay with data contained within your Google Sheet document.

Coding Mini Project Final Word Scramble Game Coding Review - project #18

Loading game data dynamically from a Google Sheet. How to share Google Sheet data for web applications. Complete review of JavaScript Scramble Game with full dynamic gameplay coming from a dynamically generated wordlist which is requested via AJAX and returned as JSON to the web application.

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript Project</title>
    <style>
        .myLink {
            font-size: 0.6em;
            text-align: center;
            display: block;
            text-decoration: none;
            padding: 5px;

        }

        input,
        button {
            width: 80%;
            margin: auto;
            display: block;
            margin-bottom: 10px;
        }

        input {
            font-size: 0.5em;
            text-align: center;
        }

        .message {
            text-align: center;
            font-size: 1em;
            color: red;

        }

        .wordOutput {
            text-align: center;
            font-size: 2em;
            letter-spacing: 0.5em;
        }

        .gameArea input {
            font-size: 1.2em;
        }

        .gameArea {
            background-color: aliceblue;
            padding: 10px;
            border: 1px solid #ddd;
        }
    </style>
</head>

<body>
    <h1>Word Scramble From Google Sheets</h1>
    <div class="output"></div>
    <div class="gameArea">
        <div class="wordOutput"></div>
        <input type="text">
        <button>Guess</button>
    </div>
    <div class="message"></div>
    <script src="pro3.js"></script>
</body>

</html>
let id = '1pKxrkT1iyzb3ELOY6vLEimn8666r8cQU6DEYFMuczB4';
const myWords = ['Hello', 'World', 'JavaScript', 'Code'];
const output = document.querySelector('.output');
const btn = document.createElement('button');

const startBtn = document.createElement('button');
startBtn.textContent = 'Start Game';
startBtn.addEventListener('click', startGame);
const game = {
    sel: '',
    scramble: '',
    wordsLeft: 0
};
const gameArea = document.querySelector('.gameArea');
const mes = document.querySelector('.message');
gameArea.style.display = 'none';
gameArea.querySelector('button').addEventListener('click', checkVal);
btn.textContent = 'load Sheet by id';
const sheetID = document.createElement('input');
sheetID.setAttribute('type', 'text');
const qs = window.location.search;
const urlParas = new URLSearchParams(qs);
const div1 = document.createElement('div');
output.append(div1);
div1.append(sheetID);
div1.append(btn);
output.append(startBtn);

btn.addEventListener('click', (e) => {
    loadListFromSheet(true);
});


if (urlParas.get('id')) {
    id = urlParas.get('id');
    //createShareLink(id);
    sheetID.value = id;
    div1.style.display = 'none';
    loadListFromSheet(false);
} else {
    sheetID.value = id;
}
```

```
function loadListFromSheet(boo) {
    console.log(sheetID.value);
    const url = 'https://spreadsheets.google.com/feeds/list/' +
sheetID.value + '/1/public/values?alt=json';
    btn.disabled = true;
    console.log(url);
    const div = document.createElement('div');
    output.append(div);
    fetch(url)
        .then(req => req.json())
        .then(json => {
            messageOut('New Word List Loaded');
            if (boo) {
                createShareLink(sheetID.value);
            }
            console.log(json['feed']['entry']);
            myWords.length = 0;
            let enty = json.feed.entry;
            enty.forEach(el => {
                console.log(el.title['$t']);
                //let temp = el['gsx$word']['$t'];
                let temp = el.title['$t'];
                console.log(temp.includes(":"));
                if (temp.length > 0 && !temp.includes(":")) {
                    let holder = temp.split(" ");
                    myWords.push(...holder);
                }
            });
            const span = document.createElement('span');
            if (boo) {
                div.append(span);
            }
            span.textContent = myWords.join(', ');
            btn.disabled = false;
            console.log(myWords);
        })
        .catch(err => {
            div.textContent = 'Error List not loaded using Default
List : ';
            messageOut('Error Loading  List');
            btn.disabled = false;
            const span = document.createElement('span');
            div.append(span);
            span.textContent = myWords.join(', ');
        })
}



function startGame() {
    gameArea.style.display = 'block';
    console.log('start game');
    output.style.display = 'none';
    console.log(myWords);
    if (myWords.length <= 0) {
        messageOut('Game Over');
        gameArea.style.display = 'none';
        output.style.display = 'block';
    } else {
        myWords.sort(() => {
            return 0.5 - Math.random()
        })
        game.sel = myWords.shift();
        game.sel = game.sel.toLowerCase();
        game.wordsLeft = myWords.length;
        game.scramble = sorter(game.sel);
        gameArea.querySelector('div').textContent = game.scramble;
    }
}

function sorter(word) {
    word = word.toLowerCase();
    let temp = word.split('');
    console.log(temp);
    temp.sort(() => {
        return 0.5 - Math.random()
    })
    temp = temp.join('');
    if (word == temp) {
        return sorter(temp);
    }
    console.log(temp);
    return temp;
}

function checkVal() {
    console.log('working');
    let guessEle = gameArea.querySelector('input');
    let guess = guessEle.value;
    guess = guess.toLowerCase();
    guessEle.value = '';
    console.log(guess);
    if (guess == game.sel) {
        messageOut('correct - words left ' + game.wordsLeft);
        startGame();
    } else {
        messageOut('incorrect');
    }
}

function messageOut(val) {
    console.log(val);
    mes.innerHTML = val;
}

function createShareLink(myId) {
    console.log(window.location.origin);
    let linkVal = window.location.origin + '?id=' + myId;
    const myURL = document.createElement('input');
    const aLink = document.createElement('a');
    aLink.textContent = "Shareable Link " + linkVal;
    aLink.classList.add('myLink');
    aLink.setAttribute('href', linkVal);
    aLink.setAttribute('target', '_blank');
    myURL.value = linkVal;
    myURL.addEventListener('focus', (e) => {
        myURL.select();
    })
    output.append(aLink);
    output.append(myURL);
}
```

Coding Mini Project Quiz Game with JavaScript and AJAX - project #19

Explore how to set up a quiz game which is flexible and dynamically generated content from JSON data. Connect to an API load JSON data and output with JavaScript to the DOM web page. Creating an interactive web game with dynamically loaded AJAX content.

Coding Mini Project Quiz Game Coding Review and Updates - project #19

JavaScript Game Project with data driven dynamically from an API endpoint. Using JavaScript creates the web application making it fully dynamic and adjustable with content. Explore how to set up a quiz game which is flexible and dynamically generated content from JSON data. Connect to an API load JSON data and output with JavaScript to the DOM web page. Creating an interactive web game with dynamically loaded AJAX content.

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript Project 4</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Noto+Serif&display=swap'
);

        body {
            font-family: 'Noto Serif', serif;
        }

        button {
            padding: 10px;
            font-size: 1.2em;
            background-color: black;
            color: white;
            cursor: pointer;
        }

        button:hover {
            opacity: 0.8;
        }

        .quez {
            font-size: 2em;
            display: block;
            border: 1px solid #ddd;
            padding: 20px;

        }

        .message {
            text-align: center;
            width: 70%;
            font-size: 2em;
            color: red;
            margin: 20px auto;

        }

        .opts {
            background-color: blue;
            margin: 5px;
            border-radius: 25px;
            padding: 15px;
        }
    </style>
</head>

<body>
    <h1>JSON</h1>
    <div class="output"></div>
    <button class="btn">Click</button>
    <script src="pro4.js"></script>
</body>
```

```
</html>
const output = document.querySelector('.output');
const btn1 = document.querySelector('.btn');
const h1ele = document.querySelector('h1');
const tempData = [];
btn1.addEventListener('click', startGame);
window.addEventListener('DOMContentLoaded', init);
btn1.disabled = true;
const game = {
    cur: 0,
    score: 0,
    gameOver: false
};
function init() {
    console.log('ready');
    //load the JSON or create the JSON data
    genQuizData();

    //document.write(JSON.stringify(tempData));

}

function genQuizData() {
    for (let ques = 0; ques < 10; ques++) {
        const holder = [];
        const ran = Math.floor(Math.random() * 3) + 2;
        for (let ops = 0; ops < ran; ops++) {
            let temp = {
                res: `Try Option Wrong ${ops+1}`,
                cor: false,
                test: ops
            };
            holder.push(temp);
        }
        let tempCorrect = {
            res: `Pick This One`,
            cor: true,
            test: 100
        };
        holder.push(tempCorrect);
        const tempObj = {
            answers: holder,
            question: `Q#${ques+1}. What is the correct answer
${ran+1}`
        }
        tempData.push(tempObj);
    }
    console.log(tempData);
    btn1.disabled = false;
    btn1.textContent = 'Start Game Quiz';
    h1ele.textContent = 'JSON dummy Data Quiz';
}


function startGame() {
    console.log('clicked');
    btn1.style.display = 'none';
    disQuestion();
}

function disQuestion() {
    //game.cur
    if (!game.gameOver) {
        output.innerHTML = '';
        h1ele.textContent = `${game.cur+1} of ${tempData.length}
Questions`;
        let question = tempData[game.cur];
        console.log(question);
        const div = document.createElement('div');
        const ques = document.createElement('h3');
        const div1 = document.createElement('div');
        ques.textContent = question.question;
        ques.classList.add('quez');
        question.answers.sort(() => {
            return 0.5 - Math.random()
        })
        question.answers.forEach((el) => {
            const sel = document.createElement('button');
            sel.classList.add('opts');
            div1.append(sel);
            sel.textContent = el.res;
            sel.addEventListener('click', (e) => {
                disButtons(div1);
                let bg = 'red';
                let mes = 'Wrong Too Bad';
                if (el.cor) {
                    console.log('Correct');
                    bg = 'green';
                    game.score++;
                    mes = 'Great you got it Right!';
                }
                sel.style.backgroundColor = bg;
                output.innerHTML += `<div class="message"
style="color:${bg}">${mes}</div>`;
            })
        })
        output.append(div);
        div.append(ques);
        div.append(div1);
    } else {
        output.innerHTML = `<h1>Game Over</h1><h2>Score: ${game.score}
out of ${tempData.length}</h2>`;
    }

}

function disButtons(ele) {
    const eles = ele.querySelectorAll('button');
    console.log(eles);
    game.cur++;

    eles.forEach((btnz) => {
        btnz.disabled = true;
    })
    btn1.style.display = 'block';
    if (game.cur >= tempData.length) {
        btn1.textContent = 'Game Over See Score';
        game.gameOver = true;
    } else {
        btn1.textContent = 'Next Question';
    }
}
```

Coding Mini Project YouTube Search API with JavaScript Connect to API   - project #20

Setup and connect to the YouTube api to search and get YouTube Data into your webpage. Dynamically create your webpage from JSON content coming from the YouTube API. Create interactive clickable content that loads from the user selection data. Use JavaScript to create a fully functional YouTube interactive web application.

Coding Mini Project YouTube Search API Output JSON data to Page - project #20

Setup and connect to the YouTube api to search and get YouTube Data into your webpage. Dynamically create your webpage from JSON content coming from the YouTube API. Create interactive clickable content that loads from the user selection data. Use JavaScript to create a fully functional YouTube interactive web application.

Coding Mini Project YouTube Search API from web page requests - project #20

Final review of YouTube API project and output to page. Coding review and updates to make the app function. Setup and connect to the YouTube api to search and get YouTube Data into your webpage. Dynamically create your webpage from JSON content coming from the YouTube API. Create interactive clickable content that loads from the user selection data. Use JavaScript to create a fully functional YouTube interactive web application.

https://developers.google.com/youtube/v3/quickstart/js

Set up your project and credentials

Create or select a project in the API Console. Complete the following tasks in the API Console for your project:

In the library panel, search for the YouTube Data API v3. Click into the listing for that API and make sure the API is enabled for your project.

In the credentials panel, create two credentials:

Create an API key You will use the API key to make API requests that do not require user authorization. For example, you do not need user authorization to retrieve information about a public YouTube channel.

Create an OAuth 2.0 client ID Set the application type to Web application. You need to use OAuth 2.0 credentials for requests that require user authorization. For example, you need user authorization to retrieve information about the currently authenticated user's YouTube channel.

In the Authorized JavaScript origins field, enter the URL http://localhost:8000. You can leave the Authorized redirect URIs field blank.

API console

https://console.developers.google.com/

Developer Docs Try the API

https://developers.google.com/youtube/v3/docs/search/list?

Example URL

https://youtube.googleapis.com/youtube/v3/search?part=snippet&maxResults=10&order=relevance&q=test&key=

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript Project 5</title>
    <style>
        * {
            box-sizing: border-box;
        }

        .output {
            display: flex;
            flex-wrap: wrap;
        }

        .search {
            flex: 100%;
            font-size: 2em;
        }

        .container {
            flex: 50%;
            font-size: 0.9em;
            text-align: center;
        }

        .box {
            padding: 5px;
            border: 1px solid black;
        }

        .box p {
            text-transform: uppercase;
        }

        .box img {
            max-width: 100%;
        }

        .btn {
            background-color: red;
            color: white;
            padding: 10px;
            font-size: 1.2em;
            display: block;
            margin: auto;
            cursor: pointer;
        }

        .btn:hover {
            opacity: 0.8;
        }

        .searchQ {
            line-height: 30px;
            font-size: 1.2em;
            width: 90%;
            margin: auto;
            display: block;
        }

        @media (max-width:680px) {
```

```
            .container {
                flex: 100%;
            }
        }
    </style>
</head>

<body>
    <h1>YouTube JSON data</h1>

    <input type="text" class="searchQ">
    <button class="btn">Search YouTube</button>
    <div class="output"></div>
    <script src="pro5.js"></script>
</body>

</html>
const apiKey = '';
const baseurl = 'https://www.googleapis.com/youtube/v3/search';
//GET
https://youtube.googleapis.com/youtube/v3/search?part=snippet&channelI
d=UCgsZ8_79Eclct_VDoql_Dwg&maxResults=10&order=relevance&q=test&key=[Y
OUR_API_KEY]

const btn = document.querySelector('.btn');
const output = document.querySelector('.output');
const searchQuery = document.querySelector('.searchQ');

btn.addEventListener('click', (e) => {
    let q = searchQuery.value || 'test';
    searchQuery.value = '';
    let paras =
'?part=snippet&channelId=UCgsZ8_79Eclct_VDoql_Dwg&maxResults=10&order=
relevance';
    let searchTerm = '&q=' + q;
    let connKey = '&key=' + apiKey;
    let url = baseurl + paras + searchTerm + connKey;
    console.log(url);
    fetch(url)
        .then(rep => rep.json())
        .then((data) => {
            console.log(data);
            output.innerHTML = `<div class="search">Search for
${q}</div>`;
            data.items.forEach(item => {
                const ele = makeCard(item);
                output.append(ele);
            });
        })
        .catch((error) => {
            console.log(error);
        })
})


function makeCard(data) {
    console.log(data);
    const vid = data.snippet;
    const main = document.createElement('div');
    main.classList.add('container');
    const div1 = document.createElement('div');
    div1.classList.add('box');
    const thumbnail = vid.thumbnails.high.url;
    const linkVideo =
`https://www.youtube.com/watch?v=${data.id.videoId}`;
    div1.innerHTML = `<p>${vid.title}</p><img src="${thumbnail}">
<div>${vid.description}</div><div>Link <a href="${linkVideo}"
target="_blank">${linkVideo}</a></div>`;
    main.append(div1);
    return main;
}
```

JavaScript JSON Post data and Mini Applications to Practice Connecting to APIs

Connecting to Endpoints using Post and Headers methods - JavaScript fetch and how to set headers and body parameters. How to create dynamically generated applications from JSON data that are interactive with AJAX requests.

Coding Mini Project Frontend API Tester connecting to Endpoints - project #21

How to make JavaScript fetch requests and return JSON data from various endpoints. Test and make interactive content that is data driven from JSON data and dynamically generate the page content with JavaScript. DOM page manipulation with JavaScript Coding.

Coding Mini Project Frontend API Tester create AJAX requests- project #21

How to create AJAX requests with JavaScript to practice API endpoints loading JSON data. How to make JavaScript fetch requests and return JSON data from various endpoints. Test and make interactive content that is data driven from JSON data and dynamically generate the page content with JavaScript. DOM page manipulation with JavaScript Coding.

Coding Mini Project Frontend API Tester loading JSON data- project #21

Testing with various methods simulating PUT and POST options for endpoint interaction. How to create AJAX requests with JavaScript to practice API endpoints loading JSON data. How to make JavaScript fetch requests and return JSON data from various endpoints. Test and make interactive content that is data driven from JSON data and dynamically generate the page content with JavaScript. DOM page manipulation with JavaScript Coding.

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript API</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Roboto&display=swap');

        * {
            box-sizing: border-box;
        }

        body {
            font-family: 'Roboto', sans-serif;
            font-size: 1.2em;
        }

        .ind {
            padding: 10px;
            width: 50px;
            display: inline-block;
            margin: 10px;
            border: 1px solid black;
            border-radius: 25px;
            text-align: center;
            cursor: pointer;
            color: white;
            background-color: #333;
        }

        .box {
            text-align: center;
            width: 80%;
            padding: 10px;
            border: 1px solid #ddd;
            margin: 10px auto;
        }

        .updater div {
            border: 1px solid #ccc;
            padding: 10px;
            background-color: azure;
            margin: 10px;
        }

        .updateBtn {
            display: block;
            width: 50%;
            margin: auto;
            text-align: center;
            padding: 10px;
            border-radius: 20px;
            font-size: 1.5em;
        }
    </style>
</head>

<body>
```

```
    <div class="container">
        <input type="text" class="val">
        <button class="btn">Click</button>
        <div class="output"></div>
    </div>
    <script src="code2.js"></script>
</body>

</html>
```

```
const btn = document.querySelector('.btn');
const inpEle = document.querySelector('input');
const output = document.querySelector('.output');
const baseurl = 'https://reqres.in/api/';
inpEle.style.display = 'none';
btn.textContent = 'Main Menu';
btn.classList.add('updateBtn');
const app = {
    pg: 1
};
//https://reqres.in/api/users?page=2

btn.addEventListener('click', loadData);

window.addEventListener('DOMContentLoaded', loadData);

function loadData() {
    const para = 'users?page=' + app.pg;
    const url = baseurl + para;
    fetch(url)
        .then(res => res.json())
        .then(data => {
            buildPage(data);
        })
}

function buildPage(data) {
    //console.log(data);
    output.innerHTML = '';
    data.data.forEach(user => {
        //console.log(user);
        const main = addUser(user);
        main.addEventListener('click', (e) => {
            userPage(user.id);
        })
    });
    const div3 = makeNode(output, 'div', '');
    div3.classList.add('box');
    for (let i = 0; i < data.total_pages; i++) {
        const span = makeNode(div3, 'span', (i + 1));
        span.classList.add('ind');
        span.addEventListener('click', (e) => {
            app.pg = i + 1;
            loadData();
        })
    }
}

function makeNode(parent, nodeType, content) {
    const el = document.createElement(nodeType);
    el.innerHTML = content;
    return parent.appendChild(el);
}

function addUser(user) {
    const output1 = makeNode(output, 'div', '');
    output1.classList.add('box');
    const html1 = `${user.first_name} ${user.last_name}`;
    const div1 = makeNode(output1, 'div', html1);
    div1.userID = user.id;
    const html2 = `${user.email}`;
    const div2 = makeNode(output1, 'div', html2);
    //const html3 = `${user.id}`;
    //const div3 = makeNode(output1, 'div', html3);
    const img1 = makeNode(output1, 'img', '');
    img1.setAttribute('src', user.avatar);
    //console.log(img1);
    return output1;
}
```

```
function userPage(id) {
    //console.log(id);
    const para = 'users/' + id;
    const url = baseurl + para;
    fetch(url)
        .then(res => res.json())
        .then(data => {
            createPage(data.data);
        })
}

function createPage(data) {
    //console.log(data);
    output.innerHTML = '';
    const main = addUser(data);
    main.classList.add('updater');
    main.setAttribute('contenteditable', true);
    const updateBtn = makeNode(main, 'button', 'update');
    updateBtn.classList.add('updateBtn');
    updateBtn.addEventListener('click', (e) => {
        const divEles = main.querySelectorAll('div');
        const userInfo = {
            "name": divEles[0].textContent,
            "id": divEles[0].userID,
            "email": divEles[1].textContent
        }
        updateUser(userInfo);
    })

}

function updateUser(userInfo) {
    console.log(userInfo);
    const para = 'users/' + userInfo.id;
    const url = baseurl + para;
    console.log(url);
    const headers = {
        "Content-Type": "application/json"
    };
    const info = {
        "name": userInfo.name,
        "email": userInfo.email
    }

    fetch(url, {
            method: "PUT",
            headers: headers,
            body: JSON.stringify(info)
        })
        .then(res => res.json())
        .then(data => {
            console.log(data);
        })
}
```

```
chrome://flags/#cookies-without-same-site-must-be-secure
```

Cookies without SameSite must be secure

If enabled, cookies without SameSite restrictions must also be Secure. If a cookie without SameSite restrictions is set without the Secure attribute, it will be rejected. This flag only has an effect if "SameSite by default cookies" is also enabled. – Mac, Windows, Linux, Chrome OS, Android

#cookies-without-same-site-must-be-secure

Coding Mini Project Get and POST Tester API JSON JavaScript - project #22

Using user input data select content from API. Send data to the server with AJAXusing JavaScript fetch both with GET and POST methods. Handle response JSON data from the server endpoint - output and generate response to the user. Get input field and form data to practice sending requests to testing endpoint which can handle both POST and GET methods returning the simulated data as a JSON object.

Lesson Tasks

1. Create input fields

2. Have user action submit the content to the server

3. Using Fetch send input field data to server

4. Handle response JSON data from the server after submission.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript API</title>
    <style>
        .box{
            display: block;
            width:80%;
            margin:10px auto;
            font-size: 1.2em;
        }
        .main{
            display: block;
            width:80%;
            margin:10px auto;
            font-size: 0.8em;
            border: 1px solid #ddd;
            padding:10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <input type="text" class="val">
        <button class="btn">Click</button>
        <div class="output"></div>
    </div>
    <script src="code3.js"></script>
</body>
</html>
```

```
const btn = document.querySelector('.btn');
const inpEle = document.querySelector('input');
const output = document.querySelector('.output');
const output1 = document.createElement('div');
output1.classList.add('main');
const baseUrl =
'https://script.google.com/macros/s/AKfycbzUcUVn99AkTK1rxxjCd-oU_707N3
s23p9OriMaMzCYunuacydj/exec';
inpEle.classList.add('box');
inpEle.setAttribute('name','nameOG');
inpEle.value = 'Hello World';
output.append(inpEle);
for(let i=0;i<10;i++){
    const myInput = document.createElement('input');
    myInput.setAttribute('type','text');
    myInput.setAttribute('placeholder','Value '+i);
    myInput.classList.add('box');
    myInput.setAttribute('name','name'+(i+1));
    myInput.value = 'Value '+i;
    output.append(myInput);
}
output.append(btn);
output.append(output1);
btn.classList.add('box');



btn.addEventListener('click', loadData);

function loadData() {
    console.log('ready');
    let url = baseUrl + '?';
    const eles = output.querySelectorAll('input');
    let tempArr = [];
    eles.forEach((el)=>{
        console.log(el.name);
        let temp = `${el.name}=${el.value}`;
        tempArr.push(temp);

    })
    let reqUrl = tempArr.join('&');
    url += reqUrl;
    console.log(url);
    getData(url);
}

function getData(url){
    fetch(url)
        .then(res => res.json())
        .then(data => {
            outputObj(data);
        })

}

function outputObj(obj){
    output1.innerHTML = '';
    for(const prop in obj){
        output1.innerHTML +=`${prop} : ${obj[prop]}<br>`;
    }
}
```

Coding Mini Project Form Submission with JavaScript fetch JSON data - project #23

Create an HTML form - submit the form input contents with AJAX to the server. Using formData create JSON from contents and send to the endpoint. Simulate form submissions with JavaScript and AJAX using GET and POST methods. Provide user detailed response and page content output from submission.

Lesson Tasks
1. Get formData using JavaScript FormData
2. Using GET method send the data to the server
3. Using POST method send the data to the server
4. Update page contents with Server Response JSON

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>JavaScript API</title>
    <style>
        .box{
            display: block;
            width:80%;
            margin:10px auto;
            font-size: 1.2em;
        }
        .main{
            display: block;
            width:80%;
            margin:10px auto;
            font-size: 0.8em;
            border: 1px solid #ddd;
            padding:10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <input type="text" class="val">
        <button class="btn">Click</button>
        <div class="output"></div>
    </div>
    <script src="code4.js"></script>
</body>
</html>
const btn = document.querySelector('.btn');
const inpEle = document.querySelector('input');
const output = document.querySelector('.output');
const myForm = document.createElement('form');
document.body.append(myForm);
const output1 = document.createElement('div');
output1.classList.add('main');
const baseUrl =
'https://script.google.com/macros/s/AKfycbzUcUVn99AkTK1rxxjCd-oU_707N3
s23p9OriMaMzCYunuacydj/exec';
inpEle.classList.add('box');
inpEle.setAttribute('name', 'nameOG');
inpEle.value = 'Hello World';
```

```
myForm.append(inpEle);
for (let i = 0; i < 10; i++) {
    const myInput = document.createElement('input');
    myInput.setAttribute('type', 'text');
    myInput.setAttribute('placeholder', 'Value ' + i);
    myInput.classList.add('box');
    myInput.setAttribute('name', 'name' + (i + 1));
    myInput.value = 'Value ' + i;
    myForm.append(myInput);
}
myForm.append(btn);
output.append(output1);
btn.classList.add('box');
btn.addEventListener('click', getPost);

function loadData(e) {
    e.preventDefault();
    console.log('ready');
    let formData = new FormData(myForm);
    let data = [...formData.entries()];
    console.log(data);
    const para = data.map(x =>
`${encodeURIComponent(x[0])}=${encodeURIComponent(x[1])}`);
    const res = para.join('&');
    console.log(res);
    let url = baseUrl + '?' + res;
    getData(url);
}

function getPost(e) {
    e.preventDefault();
    const myHeaders = new Headers();
    let formData = new FormData(myForm);
    let body = {};
    formData.forEach((val, key) => {
        body[key] = val;
    })
    console.log(body);
    const opts = {
        method: 'POST',
        myHeaders,
        body: JSON.stringify(body)
    }
    fetch(baseUrl, opts)
        .then(res => res.json())
        .then(data => {
            outputObj(data);
        })

}

function getData(url) {
    fetch(url)
        .then(res => res.json())
        .then(data => {
            outputObj(data);
        })
}

function outputObj(obj) {
    console.log(obj);
    output1.innerHTML = '';
    for (const prop in obj) {
        output1.innerHTML += `${prop} : ${obj[prop]}<br>`;
    }
}
```

Coding Mini Project JSON placeholder tester JSON JavaScript - project #24

JSON placeholder testing sending data to test server and retrieving response object as JSON. Output JSON response to the web page with JavaScript. JavaScript DOM setup and manipulate page elements outputs content dynamically using JavaScript DOM element manipulation. Great mini JavaScript and AJAX coding project to explore JSON and API sending fetch requests to a placeholder testing API.

Lesson Tasks

1. Connect to the JSON endpoint
2. Create user selected response from the JSON endpoint

3. Output JSON response to the webpage

Coding Mini Project Dynamic Content interactive CRUD tester Part 1 - project #25

Select JSON data and output to the page. Create user interaction options to create, read, update, and delete (CRUD) server side data within the testing environment. Select item by id value from server return to page. List all available posts from server output to page. Allow users to select items by id and update the content, create JSON objects to send to the server and using PUT method add to server data. Create new items for the database. Delete post item by id with user selection. Update and customize output for user from JSON data. Using JavaScript DOM manipulation update and build dashboard of dynamic content coming from the testing Server data.
Lesson Tasks
1. Create load of all endpoint posts to page
2. Allow user to select and interact with Post
3. Create new post using POST method to server
4. Add attributes and styling to page elements.
5. Make Content editable and prep to send to server

Coding Mini Project Dynamic Content interactive CRUD tester Part 2 - project #25

Lesson Tasks
1. Add PUT and DELETE methods to requests
2. Allow user to interact with Server Data and create interactive page elements
3. Update page content from JSON server Data
4. Apply styling as needed to complete the application

```
https://jsonplaceholder.typicode.com/guide/

https://my-json-server.typicode.com/

<!DOCTYPE html>
<html>

<head>
    <title>JavaScript API</title>
    <style>
```

```
        @import
url('https://fonts.googleapis.com/css2?family=Oswald&display=swap');

        * {
            box-sizing: border-box;
        }

        body {
            font-family: 'Oswald', sans-serif;
        }

        .num {
            font-size: 1.5em;
            max-width: 70px;
        }

        .box {
            border: 1px solid #ddd;
            border-radius: 25px;
            padding: 10px;
            margin: 10px auto;
            width: 80%;
        }

        .active {
            background-color: #ddd;

        }

        .del {
            padding: 5px;
            display: block;
            width: 50%;
            margin: auto;
            background-color: red;
            color: white;
            font-size: 1em;
            border-radius: 15px;
        }
```

```
.del:hover {
    opacity: 0.7;
}

.sav {
    padding: 5px;
    display: block;
    width: 50%;
    margin: auto;
    background-color: rgb(115, 255, 0);
    color: white;
    font-size: 1em;
    border-radius: 15px;
}

.container1 {
    font-size: 1.2em;
    padding: 10px;
    margin: auto;
    width: 80%;
    text-align: center;
}

.container1 input,
.container1 button,
.btn {
    font-size: 1.2em;
    padding: 10px;
    margin: auto;


}

.sav:hover {
    opacity: 0.7;
}

.active .editme {
    background-color: white;
    border: 1px solid #333;
```

```
        }
    </style>
</head>

<body>
    <div class="container">
        <input type="text" class="val">
        <button class="btn">Click</button>
        <div class="output"></div>
    </div>
    <script src="code5.js"></script>
</body>

</html>
const baseURL = 'https://jsonplaceholder.typicode.com/'
const btn = document.querySelector('.btn');
const inpEle = document.querySelector('input');
const output = document.querySelector('.output');
const container = document.querySelector('.container');
const output1 = addEle(container, 'div', '');
output1.classList.add('container1');
const btn2 = addEle(output1, 'button', 'add Item');
const input1 = addEle(output1, 'input', '');
input1.setAttribute('type', 'text');
input1.setAttribute('placeholder', 'title');
const input2 = addEle(output1, 'input', '');
input2.setAttribute('type', 'text');
input2.setAttribute('placeholder', 'body');
inpEle.value = 1;
inpEle.classList.add('num');
inpEle.setAttribute('type', 'number');
btn.textContent = 'Select Post ID';
const btn1 = addEle(output1, 'button', 'Load All Posts');
btn1.style.display = 'block';

btn.addEventListener('click', loadData);
btn1.addEventListener('click', loadAll);
btn2.addEventListener('click', addItem);
```

```
function addItem() {
    const title = input1.value || 'Title';
    const body = input2.value || 'Body Contents';
    const json = {
        title: title,
        body: body,
        userId: 1
    }

    const url = baseURL + 'posts';
    makeReq(json, 'POST', url);
}

function makeReq(json, meth, url) {

    fetch(url, {
            method: meth,
            body: JSON.stringify(json),
            headers: {
                'Content-type': 'application/json; charset=UTF-8'
            }
        })
        .then(rep => rep.json())
        .then(data => {
            addtoPage(data);
        })
}


function loadAll(e) {
    console.log('ready');
    const url = baseURL + 'posts';
    fetch(url)
        .then(rep => rep.json())
        .then(data => {
            addItems(data);
        })
}

function addItems(json) {
    output.innerHTML = '';
    json.forEach(element => {
        addtoPage(element);
    });

}

function loadData(e) {
    console.log('ready');
    const url = baseURL + 'posts/' + inpEle.value;
    fetch(url)
        .then(rep => rep.json())
        .then(data => {
            addtoPage(data);
        })
}

function addtoPage(info) {
    console.log(info);
    const main = addEle(output, 'div', '');
    const delButton = addEle(output, 'button', 'Delete Item #' +
info.id);
    delButton.classList.add('del');
    delButton.addEventListener('click', (e) => {
        fetch(baseURL + 'posts/' + info.id, {
            method: 'DELETE',
        });
        main.remove();
        delButton.remove();
    })
    main.classList.add('box');
    let html = `<h1 class="editme">${info.title}</h1>`;
    html += `<p class="editme">${info.body}</p>`;
    html += `<small>ID: ${info.id} - UserID: ${info.userId}</small>`;
    const item = addEle(main, 'div', html);
    item.addEventListener('click', (e) => {
        console.log(info.id);
        main.classList.add('active');
        const saver = addEle(main, 'button', 'Save to Server');
        saver.classList.add('sav');
        const eles = main.querySelectorAll('.editme');
        eles.forEach((el) => {
            el.setAttribute('contenteditable', true);
        })
        saver.addEventListener('click', (e) => {
            const tempObj = {
                'title': eles[0].textContent,
                'body': eles[1].textContent,
                'id': info.id,
                'userId': info.userId
            }
            eles.forEach((el) => {
                el.setAttribute('contenteditable', false);
            })
            putItem(tempObj);
            main.classList.remove('active');
            saver.remove();
        }, {
            once: true
        });
    }, {
        once: true
    })

}

function putItem(json) {
    console.log(json.id);
    console.log(json);
    const url = baseURL + 'posts/' + json.id;
    makeReq(json, 'PUT', url);
}

function addEle(parent, t, html) {
    const ele = document.createElement(t);
    parent.prepend(ele);
    ele.innerHTML = html;
    return ele;
}
```

Coding Mini Project GitHub json database custom JSON endpoint - project #26

Connect to the Massive GitHub API - create user search apply generating customized request URL with JavaScript Code. Use JavaScript to make the AJAX request for JSON data. Output and update page elements with Data from the server. Show and display JSON data values from GitHub API. Explore connecting and testing making requests to eh server output contents to the web pages with JavaScript DOM coding. Web page development and dynamic web page content. Interactive and user selected JavaScript created content driven by JSON data from the API.

Lesson Tasks
1. Create a GitHub Repo - add the db.json file
2. Create custom JSON object in GitHub
3. Connect to the endpoint using JavaScript Fetch
4. Return db.json data to the webpage and output the page elements

create your own custom web API endpoint.

```
const btn = document.querySelector('.btn');
const inpEle = document.querySelector('input');
const output = document.querySelector('.output');
const container = document.querySelector('.container');
inpEle.style.display = 'none';
btn.textContent = 'Load GitHub JSON data';

const baseurl =
'https://my-json-server.typicode.com/discoveryvip2/testingdatabase/pos
ts/';
btn.addEventListener('click', loadData);

function loadData(e) {
    console.log('ready');
    const url = baseurl;
    fetch(url)
        .then(rep => rep.json())
        .then(data => {
            createPage(data);
        })
}
```

```
function createPage(data){
    console.log(data);
    output.innerHTML = '';
    data.forEach(el => {
        const main = addEle(output, 'div', '');
        addEle(main, 'div', el.title);
        addEle(main, 'div', el.body);
        addEle(main, 'div', el.id);
        main.classList.add('box');
    });
}

function addEle(parent, t, html) {
    const ele = document.createElement(t);
    parent.append(ele);
    ele.innerHTML = html;
    return ele;
}
```

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript API</title>
    <style>
        .box{
            width:80%;
            border: 1px solid #ddd;
            margin: 10px auto;
            padding:10px;
            font-size: 1.2em;
        }
    </style>
</head>

<body>
    <div class="container">
        <input type="text" class="val">
        <button class="btn">Click</button>
        <div class="output"></div>
    </div>
    <script src="code6.js"></script>
</body>

</html>
```

Local JSON Server Node NPM JSON data Create local server for JSON data Testing

How to use Node to create a local server on your computer testing of development applications with real server responses and data saving.

Coding Mini Project Setup JSON Local Server with Node NPM - project #27

Step by step guide setup your own local testing server. Practice JavaScript AJAX requests to local data. Create your own JSON data -return data locally from json db file.   Perfect for testing and exploring making requests to your own customized JSON data endpoint.

Lesson Tasks
1. Install node https://nodejs.org/en/download/ - Open your system terminal command on windows - Mac > spotlight with CMD+Space > search terminal.
2. Check Node version and NPM version node -v npm -v
3. Create db.json file locally - add valid JSON data setup custom routes if needed as properties in the data object
4. Install json-server on your system using the terminal npm install -g json-server (prefix with sudo if on a mac and you encounter permissions issue.)
5. Run the JSON server package - json-server --watch db.json - Open your web browser to http://localhost:3000/posts

Coding Mini Project Connect to Local Server Get and Post Requests with Fetch - project #28

Practice connecting to JSON data locally. Setup your own JSON data using JavaScript to connect to the local JSON data and output it to your testing dev app. Javascript local web developer environment setup and practice environment. Get JSON data use JavaScript to update page elements.

Lesson Tasks

1. Connect to Local JSON file
2. Get all the posts in the JSON file from the endpoint
3. Output the response JSON to the console
4. Create an option for the user to post to the database adding to the json file from the input field

Practice connecting and making POST and GET requests to local JSON server using a json.db file located locally. Create page elements with JavaScript using Data returned from JSON data object.

Coding Mini Project - Connect to Local Server Put and Delete Requests with Fetch project #29

Lesson Tasks
1. Output Posts to Page
2. Create event listeners to update and Delete page posts by id
3. Send fetch PUT and DELETE methods to JSON server
4. Add styling debug application as needed.

Practice connecting and making PUT and DELETE requests to local JSON server using a json.db file located locally. Create page elements with JavaScript using Data returned from JSON data object. Update Manipulate JSON local data simulating Database requests to the backend code.

```
<!DOCTYPE html>
<html>

<head>
    <title>JavaScript API</title>
    <style>
        * {
            box-sizing: border-box;
        }

        .container {
            text-align: center;
            margin: auto;
        }

        .message {
            font-size: 1em;
            color: #222;
            padding: 20px;
            border: 1px solid #ddd;
        }

        .loadPosts {
            display: block;
            width: 50%;
            font-size: 1.5em;
            background-color: blue;
            color: white;
            margin: auto;
        }

        .dashboard button {
            color: white;
            font-size: 1em;
            border-radius: 15px;
        }

        .dashboard button:hover {
            opacity: 0.7;
        }

        .dashboard button:nth-child(1) {
            background-color: green;
        }

        .dashboard button:nth-child(2) {
            background-color: red;
        }

        .box {
            display: block;
            width: 70%;
            border: 1px solid #ddd;
            margin: 5px auto;
            padding: 10px;
        }

        .box input {
            width: 90%;
```

```
            display: block;
            padding: 3px;
            margin: 5px auto;
            font-size: 1.2em;
        }

        .box button {
            padding: 10px;
            width: 40%;
            cursor: pointer;
        }

        .selInfo {
            width: 30%;
            display: inline-block;
            height: 40px;
            font-size: 1.5em;
        }
    </style>
</head>

<body>
    <div class="container">
        <input type="text" class="val">
        <button class="btn">Click</button>
        <div class="output"></div>
    </div>
    <script src="apps1.js"></script>
</body>

</html>
const container = document.querySelector('.container');
const message = cme2(container, 'div', 'Message Area');
message.classList.add('message');
const val1 = document.querySelector('.val');
const output = document.querySelector('.output');
const baseurl = 'http://localhost:3000/';
const btn1 = document.querySelector('.btn');
const btn2 = cme2(container, 'button', 'Load Posts');
btn2.classList.add('loadPosts');
```

```
btn1.textContent = 'Create New';
btn1.classList.add('selInfo');
val1.classList.add('selInfo');
window.addEventListener('DOMContentLoaded', getAllPosts);
btn1.addEventListener('click', addPost);
btn2.addEventListener('click', getAllPosts);

function addPost(e) {
    console.log('ready');
    e.preventDefault();
    const title = val1.value || 'default title';
    val1.value = '';
    const url = baseurl + 'posts';
    const body = {
        title: title,
        author: 'Laurence Svekis'
    }
    const opts = {
        method: 'POST',
        body: JSON.stringify(body),
        headers: {
            'Content-type': 'application/json; charset=UTF-8'
        }
    };
    fetch(url, opts)
        .then(rep => rep.json())
        .then(data => {
            makeItem(data);
            myMessage(`New Item Made id = ${data.id}`);
        })
}

function myMessage(html) {
    message.innerHTML = html;
}

function getAllPosts(e) {
    console.log('page ready');
    const url = baseurl + 'posts';
    fetch(url)
```

```
        .then(rep => rep.json())
        .then(data => {
            pageContents(data);
        })
}



function pageContents(data) {
    console.log(data);
    output.innerHTML = '';
    data.forEach(el => {
        makeItem(el);
    });
}


function makeItem(el) {
    console.log(el);
    const main = cme(output, 'div', '');
    main.classList.add('box');
    const myID = cme(main, 'div', el.id);
    const in1 = cme(main, 'input', '');
    in1.value = el.title;
    const in2 = cme(main, 'input', '');
    in2.value = el.author;
    const btns = cme(main, 'div', '');
    btns.classList.add('dashboard');
    const bt1 = cme(btns, 'button', 'Update');
    const bt2 = cme(btns, 'button', 'Delete');
    bt1.addEventListener('click', (e) => {
        const json = {
            title: in1.value,
            author: in2.value
        };
        updateItem(json, el.id);
        myMessage(`Updated = ${el.id}`);
    })
    bt2.addEventListener('click', (e) => {
        myMessage(`Deleted = ${el.id}`);
        const url = baseurl + 'posts/' + el.id;
        fetch(url, {
            method: 'DELETE'
        });
        main.remove();
    })
}

function updateItem(json, id) {
    const url = baseurl + 'posts/' + id;
    const opts = {
        method: 'PUT',
        body: JSON.stringify(json),
        headers: {
            'Content-type': 'application/json; charset=UTF-8'
        }
    };
    fetch(url, opts)
        .then(rep => rep.json())
        .then(data => {
            console.log(data);
        })
}

function cme2(parent, typeEle, html) {
    const el = document.createElement(typeEle);
    parent.prepend(el);
    el.innerHTML = html;
    return el;
}

function cme(parent, typeEle, html) {
    const el = document.createElement(typeEle);
    parent.append(el);
    el.innerHTML = html;
    return el;
}


{
```

```
"posts": [
    {
        "title": "UPDATE",
        "author": "Helloworld",
        "id": 1
    },
    {
        "title": "new bew",
        "author": "Laurence Svekis",
        "id": 12
    },
    {
        "title": "title",
        "author": "Laurence Svekis",
        "id": 13
    },
    {
        "title": "test",
        "author": "Laurence Svekis",
        "id": 14
    },
    {
        "title": "test",
        "author": "Laurence Svekis",
        "id": 15
    },
    {
        "title": "default title",
        "author": "Laurence Svekis",
        "id": 16
    },
    {
        "title": "Hello World 2",
        "author": "Laurence Svekis",
        "id": 17
    }
],
"comments": [
    {
        "id": 1,
            "body": "some comment",
            "postId": 1
        }
    ],
    "profile": {
        "name": "typicode"
    },
    "tester": {
        "val": 1
    }
}
```

AJAX Contact form submit data to Google Sheet and Send emails from Google Apps Script MailApp Service.

How to send an email from client side code AJAX Contact form submit data to Google Sheet and Send emails from Google Apps Script MailApp Service

Coding Mini Project - JavaScript AJAX web contact form sending Emails with Apps Script project #30

How to send an email from client side code AJAX Contact form submit data to Google Sheet and Send emails from Google Apps Script MailApp Service. Create your own custom fully functional web contact form that sends emails, tracks form content into a spreadsheet. Setup your own endpoint using Google Apps Script and creating a fully functional web app. Send POST requests from your web page using AJAX and JavaScript fetch - create JSON structure of web form data from inputs. Get web page form fields, create JSON object.

Lesson Tasks
1. Create contact for with HTML or JavaScript
2. Select form add submit event listener
3. Get form fields names and values
4. Apply conditions for data from form
5. Check valid submission data and create JSON object to send with POST method to endpoint. Prepare fetch request for endpoint.

Coding Mini Project - Web App Endpoint Setup Google Apps Script project #30

Lesson Tasks
1. Login into your Google Account
2. Create a new Sheet doc access the script editor from the sheet
3. Create a doPost(e) method, get the **e.postData.contents** and parse in javascript object
4. Connect to sheet appendRow of contents
5. Use **MailApp.sendEmail()** to send an email to your account.

Coding Mini Project - JavaScript AJAX web contact Form testing and final code review Project #30

Lesson Tasks
1. Test application ensure its sending the data from the form
2. apply Styling as needed
3. Provide user messaging for the status of the sent data

Many Google apps, one platform in the cloud

Increase the power of your favorite Google apps — like Calendar, Docs, Drive, Gmail, Sheets, and Slides.

Apps Script lets you do more with Google, all on a modern JavaScript platform in the cloud. Build solutions to boost your collaboration and productivity.

https://developers.google.com/apps-script

Drive https://drive.google.com/drive/my-drive

Permissions remove https://myaccount.google.com/permissions

```
const myForm = document.querySelector('#signup');
const output = document.querySelector('.output');
output.style.display = 'none';
const url =
'https://script.google.com/macros/s/AKfycbzGo****XAnGx9m7A76nI0SM/exec
';
myForm.addEventListener('submit', (e) => {
    e.preventDefault();
    const ele = myForm.elements;
```

```
//console.log(ele);
//console.log('Sending Data');
const holder = {};
const err = [];
for (let i = 0; i < ele.length; i++) {
    //console.log(ele[i]);
    const el = ele[i];
    let val = true;
    if (el.getAttribute('type') == 'submit') {
        val = false;
    };
    if (el.name == 'user') {
        if (el.value.length < 5) {
            val = false;
            err.push('Name needs to be 5 or more');
        }
    }
    if (el.name == 'email') {
        let check = validateEmail(el.value);
        console.log(check);
        if (!check) {
            val = false;
            err.push('Not valid email');
        }
    }
    if (val) {
        holder[el.name] = el.value;
    }
}
if (err.length > 0) {
    output.innerHTML = '';
    output.style.display = 'block';
    err.forEach(error => {
        output.innerHTML += '<div> ' + error + '</div>';
    })
} else {
    //form submit
    console.log(holder);
    output.style.display = 'block';
    output.innerHTML = 'Sending...'
```

```
        fetch(url, {
                method: 'POST',
                body: JSON.stringify(holder)
            })
            .then(rep => rep.json())
            .then(data => {
                console.log(data);
                clearForm();
            })
    }
})

function clearForm() {
    const ele = myForm.elements;
    output.style.display = 'none';
    output.innerHTML = '';
    for (let i = 0; i < ele.length; i++) {
        if (ele[i].getAttribute('type') != 'submit') {
            ele[i].value = '';
        }
    }
}

function validateEmail(email) {
    const re = /\S+@\S+\.\S+/;
    return re.test(email);
}


<!DOCTYPE html>
<html>

<head>
    <title>JavaScript API</title>
    <style>
        .container {
            width: 80%;
            margin: auto;
            background-color: #ddd;
            padding: 15px;
            border-radius: 25px;
        }

        input,
        textarea {
            display: block;
            width: 100%;
            margin: 10px 0;
            font-size: 1.2em;
            border-radius: 5px;
            padding: 5px;
        }

        input[type="submit"] {
            background-color: black;
            color: white;
            cursor: pointer;
        }

        input[type="submit"]:hover {
            opacity: 0.8;
        }

        .output {
            background-color: white;
            padding: 25px;
            text-align: center;
            border: 1px solid red;
            font-family: Impact, Haettenschweiler, 'Arial Narrow
Bold', sans-serif;

        }

        .output div {
            margin: 5px 0;
        }
    </style>
</head>

<body>
```

```
    <div class="container">
        <form id="signup">
            <div>
                <input type="text" name="user" value="Laurence"
required>
            </div>
            <div>
                <input type="text" name="email"
value="gappscourses@gmail.com" required> </div>
            <div>
                <textarea name="message">Hi How are You</textarea>
</div>
            <div>
                <input type="submit" value="Send Message"> </div>
        </form>
        <div class="output"></div>
    </div>
    <script src="code7.js"></script>
</body>

</html>
function doGet(e){
  Logger.log('hello');
  const obj = {
    parameter : 'test'
  }
  const output = JSON.stringify(e);
  console.log(output);
  const test = ContentService.createTextOutput(output);
  return test;
  //return ContentService.createTextOutput(JSON.stringify(e));
}

function doPost(e){
  const json = JSON.parse(e.postData.contents);
  const email = json.email;
  const user = json.user;
  const message = json.message;
  let valid = true;
  if(!validateEmail(email)){
    valid = false;
  }
  const id = '1LH******6phAOyeb8yEbr7HlNS8';
  const ss = SpreadsheetApp.openById(id);
  const sheet = ss.getSheetByName('Sheet1');
  //sheet.appendRow([e.postData.contents]);
  if(valid){
    mailer(email,user,message);
    sheet.appendRow([email,user,message,'Message Sent']);
  }
  sheet.appendRow([email,user,message,valid]);
  return ContentService.createTextOutput(JSON.stringify(json));
}
function mailer(email,user,message){
  MailApp.sendEmail({
    to: 'gappscourses@gmail.com',
    subject : 'From Web Form',
    htmlBody: 'From Name : ' + user + '<br>Email  '+ email +
'<br>Message ' + message
  });
}
function tester(){
  mailer('test','user','message');
}

function validateEmail(email){
    const re = /\S+@\S+\.\S+/;
    return re.test(email);
}
```