JavaScript Dynamic Quiz with Sheets Data

Create a Spreadsheet get the Data **5**

Create a Web App of Data **11**

Get Sheet data with JavaScript **15**

Google Apps Script Data Object **20**

JavaScript Dynamic Quiz from Sheet Data **23**

Submit content to Sheet POST **32**

Complete Quiz with Score Submission to Google Sheets **39**

How to create an API from Google Sheets Data

For the section you will need to have a Google Account, it can be a free Google Account or an organizational Google Account. To create Google Sheets and create an Apps Script application you will need to log into your Google account.

Explore how to create a Quiz Application and output Google Spreadsheet data as JSON for an API. This section will demonstrate how to output Spreadsheet data as JSON in a web app with Google Apps Script and Connecting to the endpoint from the webapp using JavaScript Code.

You can get more information about Google Apps Script at https://www.google.com/script/start/

Google Apps Script provides developers a way to access Google Workspace services and more with code. Google Apps Script is a coding language based on JavaScript that runs in the cloud. It allows you to interact with the Google Suite of products, and create customized functionality within those products. Google Apps Script , Google's server side Javascript-based cloud scripting platform for automating tasks across Google products and third-party services is available and can be accessed with an Google Account.

You will need a Google Account in order to access the Google Suite. You can sign up for free at Google or if your company has an organizational account with Google Workspace you can use that as well. There are some differences on permissions and who can access the documents between the organization accounts and the regular Google accounts.

Log into your Google Account and in the script dashboard create a new project. This will open the project in the Google Apps Script Editor.
https://script.google.com/

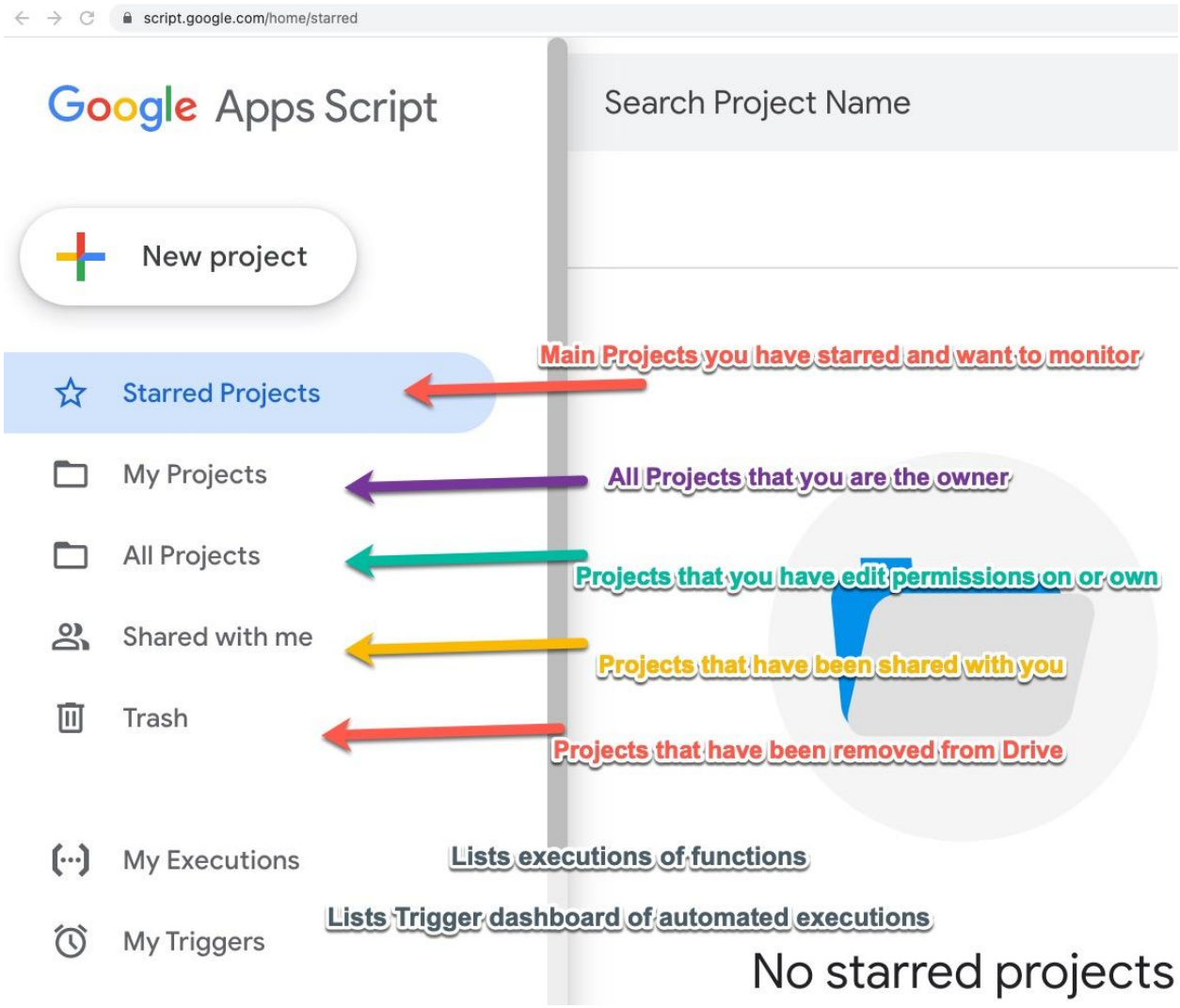


Illustration of Google Apps Script Editor Menu Items in sidebar

By default the Apps Script editor will create a file called Code.gs and create a empty function in the editor called function myFunction(){}

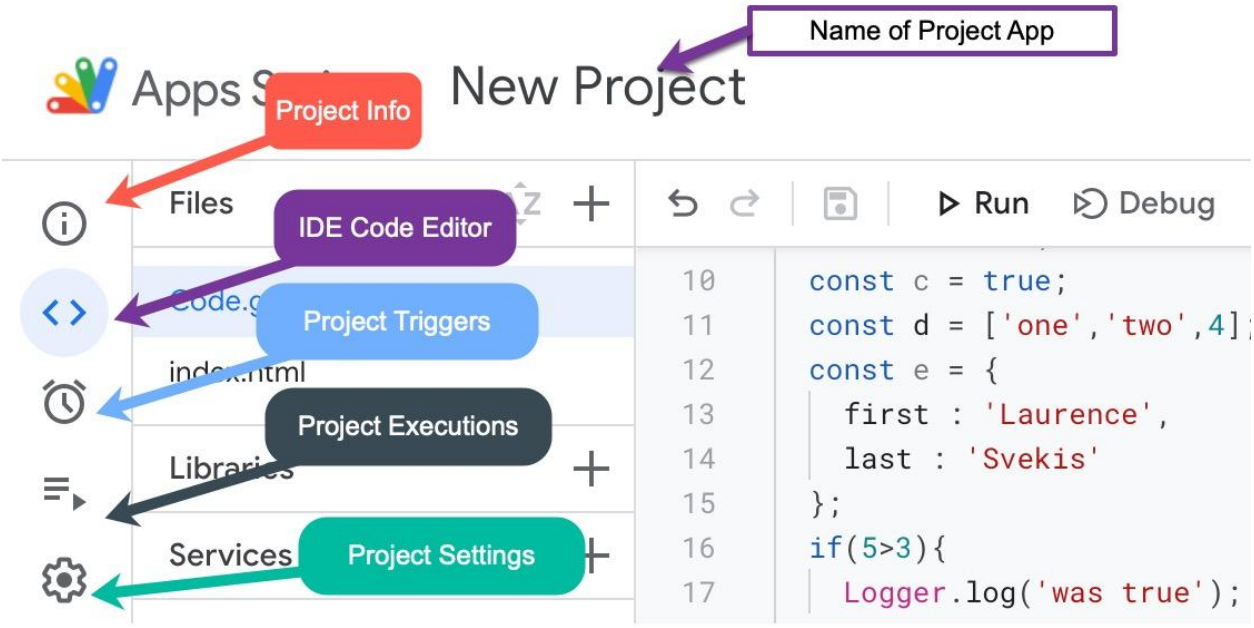


Illustration of Google Apps Script Editor Menu Items navigation of sidebar

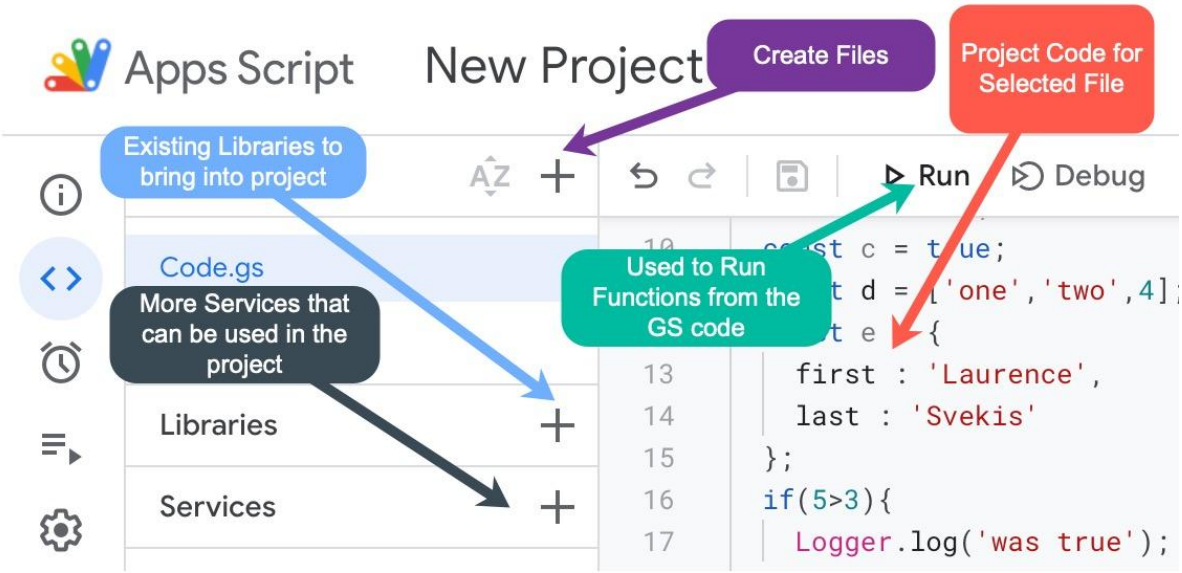


Illustration of Google Apps Script Editor Menu quick menu bar and IDE

The Logger is a built-in function in Apps Script that allows you to send data to be displayed in the execution log. The execution log will display once the function is run if the Logger function is called. This log is lightweight and streams in real time,

but persists only for a short time and can be used for error checking, debugging and retrieving information as the code runs.

In the code editor you can write regular JavaScript code, as Apps Script and JavaScript code runs within a runtime or runtime environment containing the JavaScript engine that parses and executes script code.

Create a Spreadsheet get the Data

To create a Spreadsheet you can log into your Google Account and select go to the drive and select the New button in the top left hand corner.

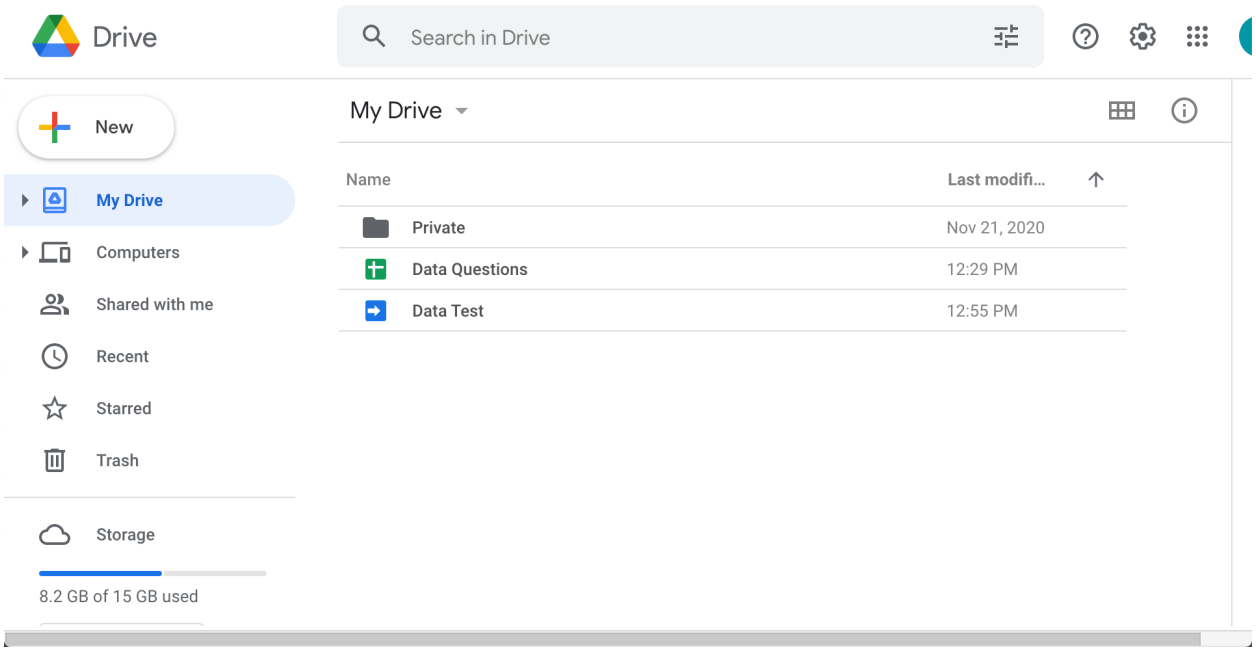


Illustration of Google Drive and how to create a New Spreadsheet

Create a new spreadsheet and add data to the spreadsheet that you want to output into the web app endpoint.

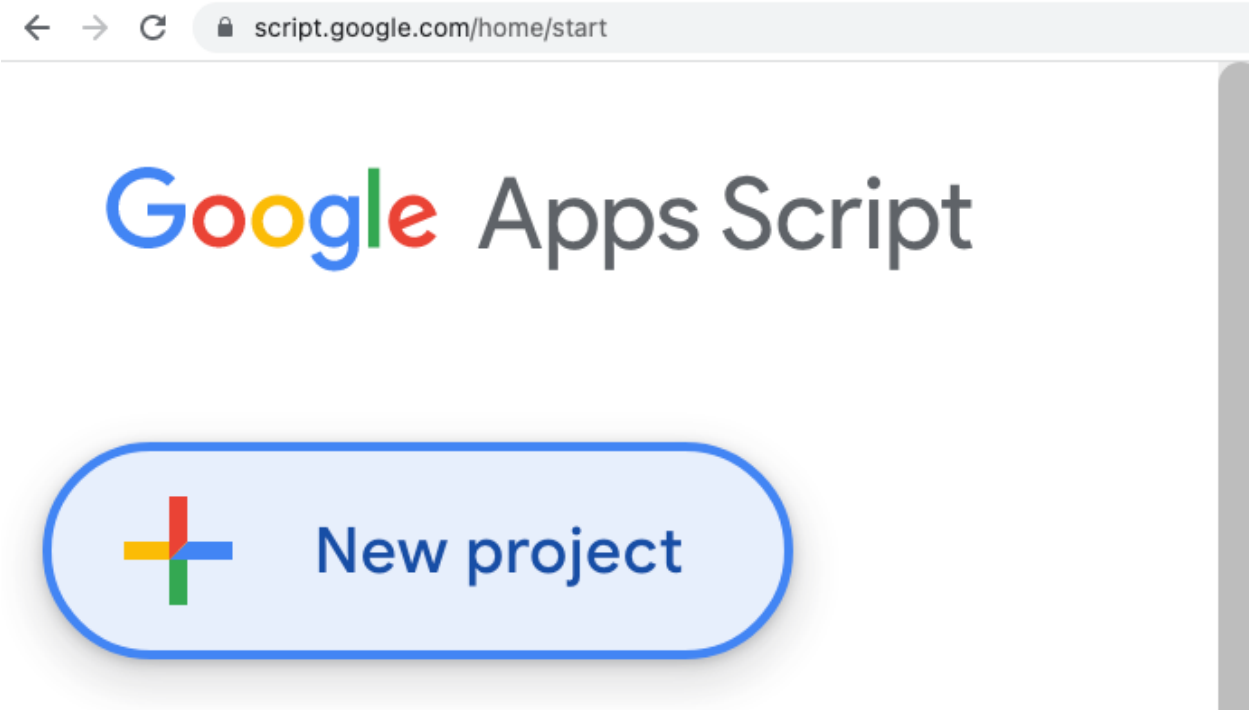
You can create your own data or use the table of data from below.

Question	Correct	Option 1	Option 2	Option 3	Option 4	Option 5
What is 5 + 10	15	4	14	25	33	
Do you like JavaScript	Yes	No				
How many wheels on a car	4	3	5	7		

Within the browser bar, select the unique id for your spreadsheet file. Each Google file has a unique identifier that can be used within the code to select the file object and its contents.

Spreadsheet ID in the browser address bar

Create a new Apps Script file, selecting a new project in the scripts dashboard menu from the top left side drop down menu. https://script.google.com/home/start



Create a new project with the New project button.

Give the Apps Script project a name by clicking the name at the top of the editor.

Data Test is the name for this script.

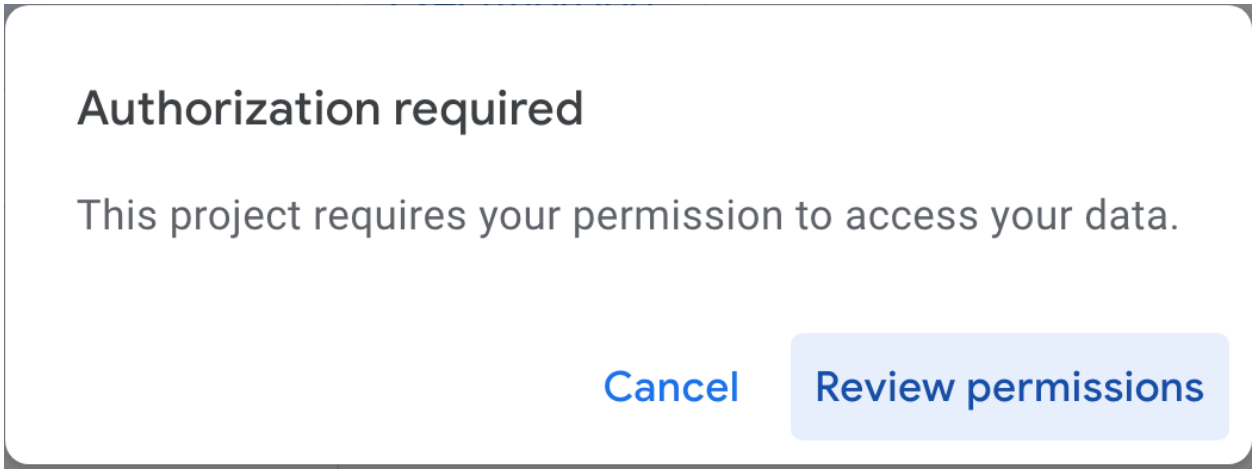
Top quick menu bar with Run and Debug options, will let you select the function you want to run from the drop down. Once you select a function you can then press run to run the code within the function.

Quick menu bar for running functions from the drop down selector

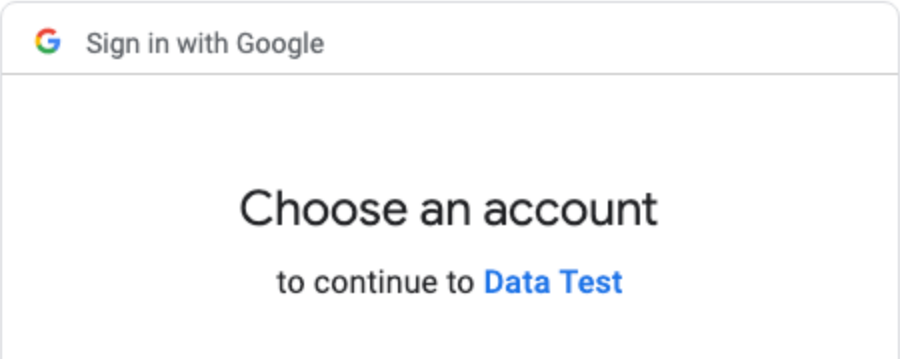
When the script needs to access new permissions, the permissions screen will popup. In order for the script to interact with the workspace permissions need to be accepted. They can also be removed if needed at https://myaccount.google.com/permissions and selecting the Apps Script by its name.

Accepting permissions :

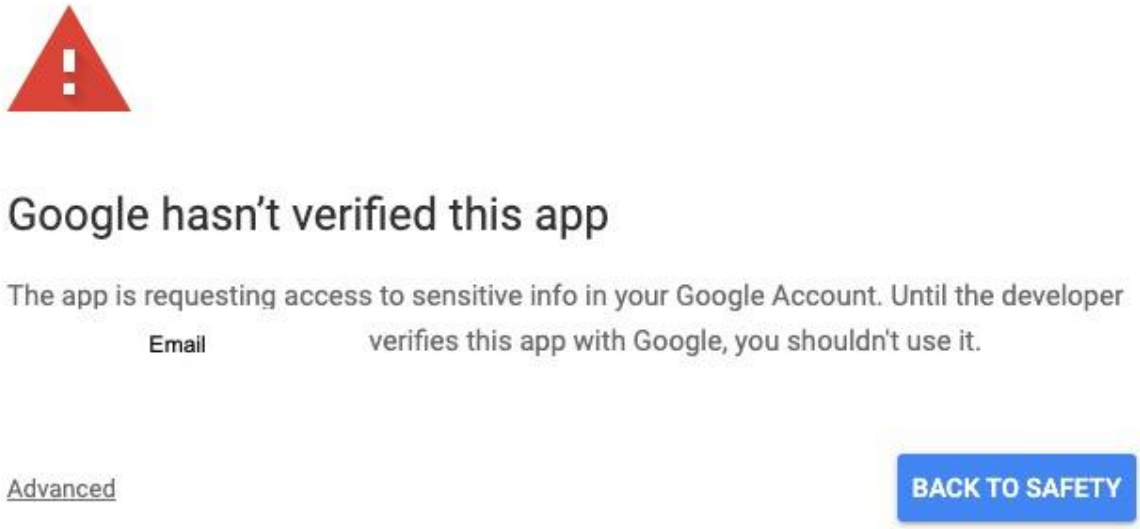
1. Authorization popup - Select "Review permissions".



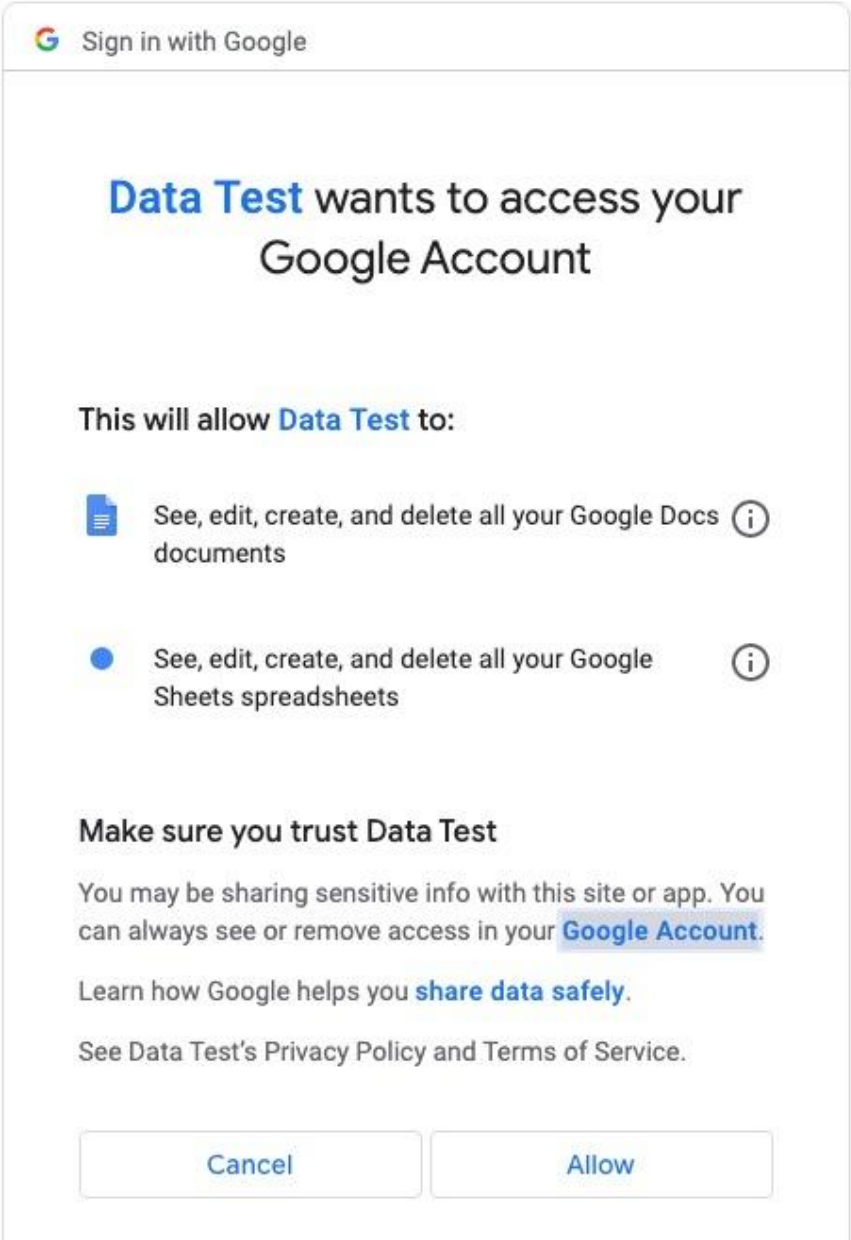
2. Select the Google account that will be used for the script.



3. Google warning message that it has not verified the app. Select “Advanced” and select the app name to move to the next screen.



4. Permissions Screen with which permissions are being used by the app. To accept, press the Allow button at the bottom.



Exercise - Create a Google Apps Script script that will get the Spreadsheet Data and output it into the execution logger.

1. Create a function that you can run in the Apps Script editor.
2. Using the unique id of the sheet, and the SpreadsheetApp class, open the spreadsheet as an object assigned to a variable name. const ss = SpreadsheetApp.openById(id);

3. Select the sheet you want to use from the spreadsheet object, const sheet = ss.getSheetByName('questions');
4. Select the range of cells to use from the sheet object, const range = sheet.getDataRange();
5. Select the values from the selected range, const data = range.getValues();
6. The returned result will be the array of data from the spreadsheet in a nested array. Each row is a separate array item in the parent array.

```
[[Question, Correct, Option 1, Option 2, Option 3, Option 4, Option 5], [What is 5 +
10, 15.0, 4.0, 14.0, 25.0, 33.0, ], [Do you like JavaScript, Yes, No, , , , ], [How
many wheels on a car, 4.0, 3.0, 5.0, 7.0, , ]]
```

Sample of array of data from the spreadsheet.

```
function getSheetInfo() {
 const id = '1S*******Gu39pi0';
 const ss = SpreadsheetApp.openById(id);
 //console.log(ss);
 //Logger.log(ss);
 const sheet = ss.getSheetByName('questions');
 //Logger.log(sheet);
 const range = sheet.getDataRange();
 //Logger.log(range);
 const data = range.getValues();
 //Logger.log(data);
 data.forEach(row =>{
   Logger.log(row);
 })
 const dd = DocumentApp.create('test');
}
```

Create a Web App of Data

Web apps can be deployed in Google Apps Script , which allow us to create a web URL for the returned content. This then can be used as an endpoint to connect to with a web application, and retrieve the data from the web app. The web app in Google Apps Script provides a way to connect to the Spreadsheet data, and output the spreadsheet data into a web URL.

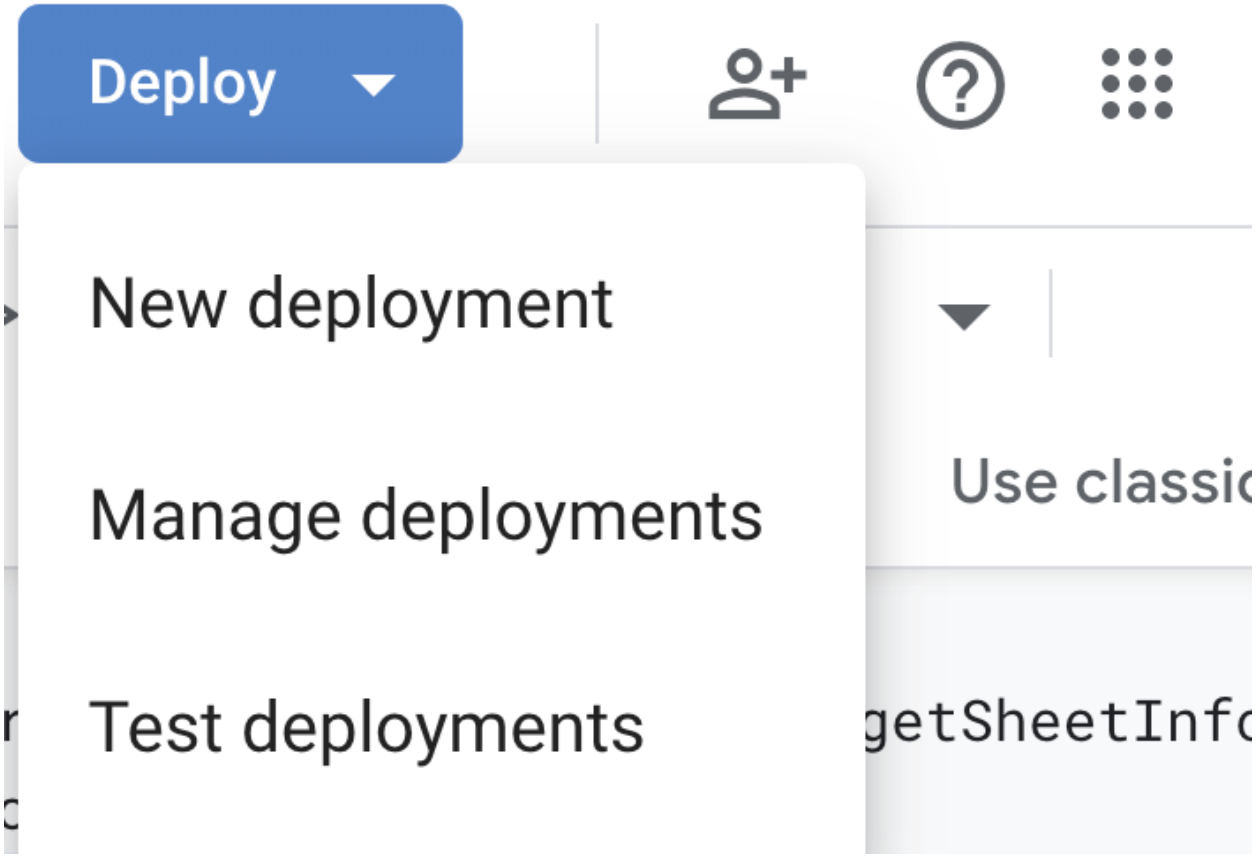
To create a web app, the Apps Script code will need to use the doGet() function. This function is used by the web app, and the output for the web app comes from the returned contents of the doGet() function. There are several classes in Google Apps Script that can be used, the Class ContentService is the service for returning text content from a script. Properties of the ContentService that can be used are MimeType and it also includes the method createTextOutput() which can create a new TextOutput object.

To deploy a new web app, and get the URL for the endpoint :

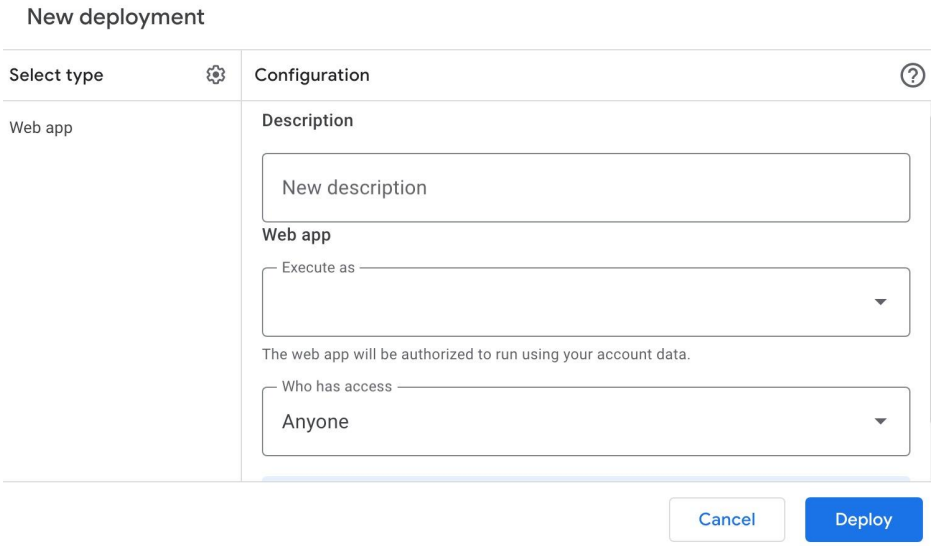
1. Select the “Deploy” button at the top of the editor.

Deploy button in Apps Script Editor

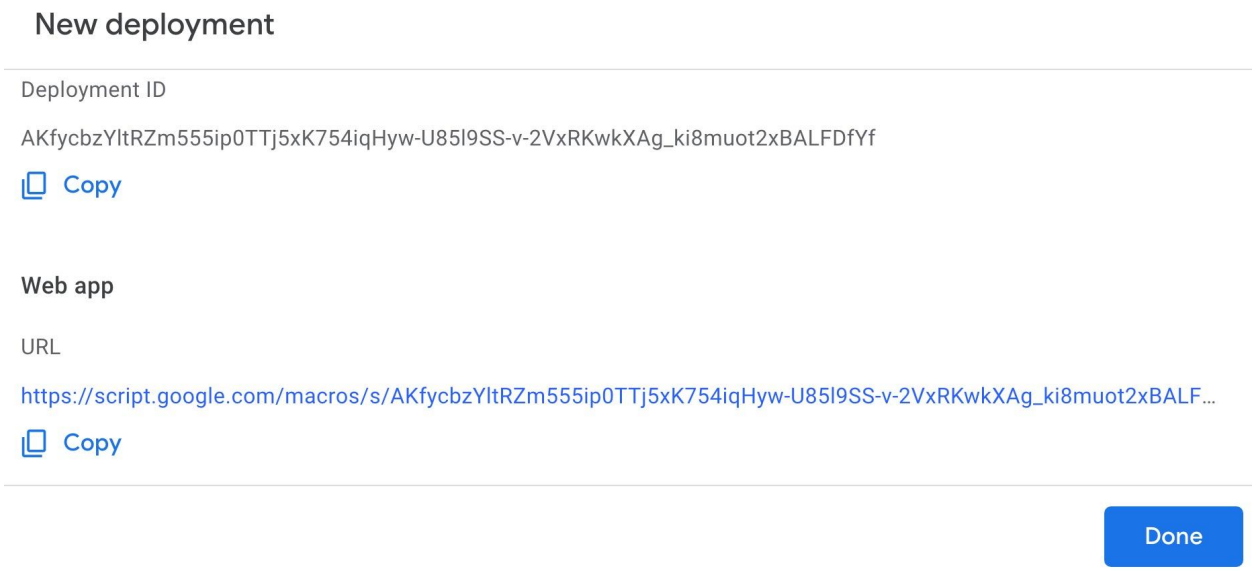
2. Select from the drop down “New Deployment”



3. Select from the cog on the Select Type, a web app. Then enter the configuration details. The "description" should be set to info about what the app will do, the "execute as" should be the Apps Script developer account, and the "who has access" should be set to anyone which will allow all users with the URL to access the content.



4. Select the Deploy button and the next screen will provide the web app URL. Copy the URL or click to open it in the browser.



Exercise : Create a web app with the Sheet Data as JSON

1. Create the doGet(e) function
2. Using the code from the previous lesson, return the sheet data as a nested array of content.
3. Use the JSON.stringify() to convert the array into a string value.
4. Return the value of the stringified sheet data, using the ContentService.createTextOutput() method.
5. Set the MimeType as JSON data setMimeType(ContentService.MimeType.JSON)
6. Deploy the web app, visit the URL and check for the correct values in the JSON output to match your sheet data.

```
function doGet(e){
 const output = {
   first : 'Laurence',
   last : 'Svekis'
 }
 const strOutput = JSON.stringify(getSheetInfo());
```

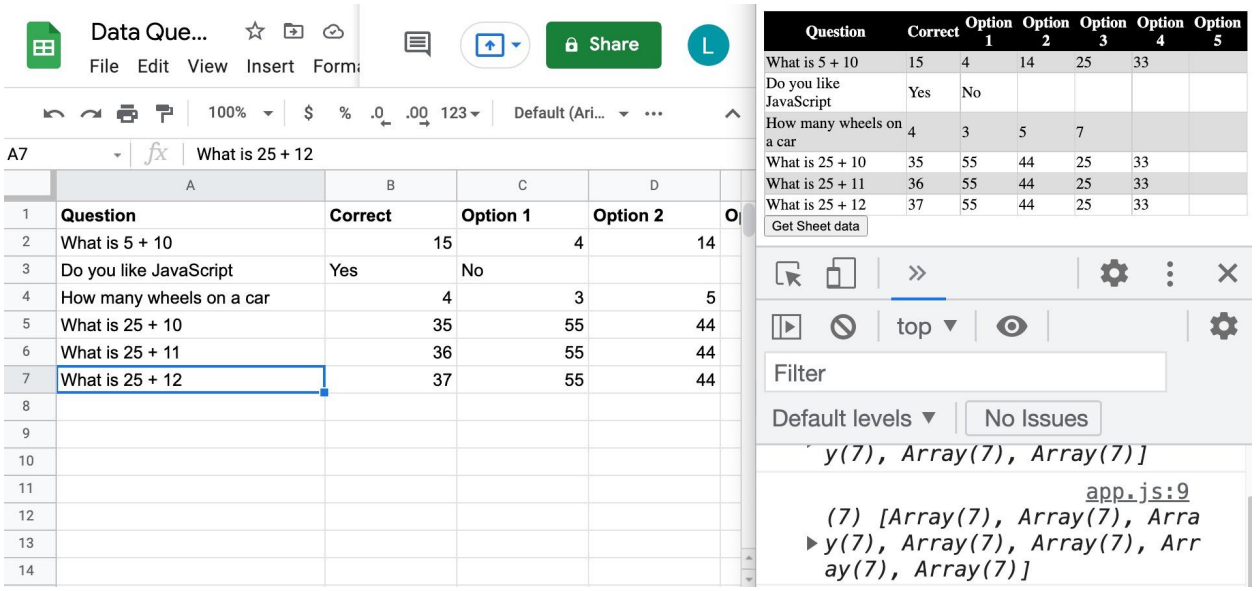
```
 Logger.log(strOutput);
 return
ContentService.createTextOutput(strOutput).setMimeType(ContentSe
rvice.MimeType.JSON);
}


function getSheetInfo() {
 const id = '1SOufD8********Gu39pi0';
 const ss = SpreadsheetApp.openById(id);
 const sheet = ss.getSheetByName('questions');
 const range = sheet.getDataRange();
 const data = range.getValues();
 Logger.log(data);
 return data;
}
```

Get Sheet data with JavaScript

The web app is outputting the Sheet data as an array, with rows as nested arrays. Using JavaScript and fetch, the web app URL can be used as an endpoint and the data that is JSON structured can be brought into the JavaScript code. This will be returned as an array, which then can be looped through to get the content. The first row will contain the headings from the sheet's first row of content.

The sheet data is an array with nested rows for each row of contents in the Sheet. Fetch will return the sheet contents as below.
[["Question","Correct","Option 1","Option 2","Option 3","Option 4","Option 5"],["What is 5 + 10",15,4,14,25,33,""],["Do you like JavaScript","Yes","No","","","","",""],["How many wheels on a car",4,3,5,7,"",""],["What is 25 + 10",35,55,44,25,33,""],["What is 25 + 11",36,55,44,25,33,""],["What is 25 + 12",37,55,44,25,33,""]]

Content updates on the Spreadsheet, will immediately be able to be accessed using the AJAX request from JavaScript. The endpoint data will update and the returned data from the request will have the new updated values.



Exercise : Create a table of Spreadsheet data in the web page HTML elements.

1. Create the HTML for the application, you can use the default HTML file code from below or customize your own styling and elements.
2. Assign a value of the Google Apps Script WebApp URL to a variable of url.
3. Select the page elements as objects
4. Add an event listener to the button when clicked fetch the contents of the web URL
5. Using fetch, get the endpoint data as JSON, slice the rows using slice(1) array method, and get the first row of headings data. Send the data into a function named outputData()
6. Create a function named maker() that can create page elements, with parameters of the element tag type, the element parent and the class for the element.
7. In the outputData() function create a main table, add a row for th cells
8. Loop through the headings from the array data, create th element cells and add the contents of the array data to the cell textContent.
9. Loop through the rows of arrays, representing the rows from the Spreadsheet. Create a main row for the cells.
10. Iterate through the columns, from the row data. Create a new td element that will contain the col data from the row.
11. Apply CSS as needed and desired for the output.

HTML

```
<!doctype html>
<html>
<head>
  <title>JavaScript JSON AJAX</title>
  <style>
     .td{
        border:1px solid #ccc;
     }
```

```
      .th{
         background-color:black;
         color:white;
      }
      .row:nth-child(even){
         background-color:#ddd;
         color:black;
      }
      .table{
         width:100%;
         border:1px solid black;
         border-collapse :collapse;
      }
   </style>
</head>
<body>
   <div>
      <div class="output"></div>
      <button class="btn">Click Me</button>
   </div>
   <script src="app.js"></script>
</body>
</html>
```

JAVASCRIPT

```
const url = 'https://script.google.com/macros/s/AKf**I4Mk/exec';
const btn = document.querySelector('.btn');
const output = document.querySelector('.output');
```

```
btn.textContent = 'Get Sheet data';
btn.onclick = (e)=>{
   fetch(url)
   .then(rep => rep.json())
   .then(data =>{
       console.log(data);
       const headings = data[0];
       const rows = data.slice(1);
       output.innerHTML = JSON.stringify(data);
       outputData(headings,rows);
   })
}

function outputData(headings,rows){

   const tabl = maker('table',output,'table');
   const heading = maker('tr',tabl,'row');
   headings.forEach(ele =>{
       const el = maker('th',heading,'th');
       el.textContent = ele;
   })
   rows.forEach(row =>{
       const r = maker('tr',tabl,'row');
       row.forEach(col =>{
           const el = maker('td',r,'td');
           el.textContent = col;
       })
   })
```

```
}

function maker(t,parent,c){
   const el = document.createElement(t);
   el.classList.add(c);
   return parent.appendChild(el);
}
```

Google Apps Script Data Object

The endpoint JSON data can be better set up for the application, and this can be setup with Google Apps Script before the JSON is output to the endpoint URL. Typically on the server side for the API the data will be structured in the format needed by the application. The server side code is best suited to create the JSON object structure with objects and arrays. This will make it easier with meaningful object key names, that will make the data more readable when a web developer is connecting to the endpoint.

The data will go from nested arrays to an array with iterable object items.

[{"question":"What is 5 + 10","answer":15,"opts":[15,4,14,25,33]},{"question":"Do you like JavaScript","answer":"Yes","opts":["Yes","No"]},{"question":"How many wheels on a car","answer":4,"opts":[4,3,5,7]},{"question":"What is 25 + 10","answer":35,"opts":[35,55,44,25,33]},{"question":"What is 25 + 11","answer":36,"opts":[36,55,44,25,33]},{"question":"What is 25 + 12","answer":37,"opts":[37,55,44,25,33]}]

Exercise : Using the Spreadsheet data output the JSON as more readable JSON

1. Create a function that gets the spreadsheet data as a variable data.
2. Using the array slice method, remove the first row of heading data from the array.
3. Using JavaScript array map method, loop through all the row items in the data array.
4. For each row array, get the first column value as a question, and the second column value as the correct answer. Create a temporary object, add an array named opts for all the possible answers to the question.
5. Loop through the items in columns with index values 1-6, if the cell has a value then add it into the opts array within the temporary object. Return the temporary array in the map method.
6. Return the new object array for values from the Sheet data.

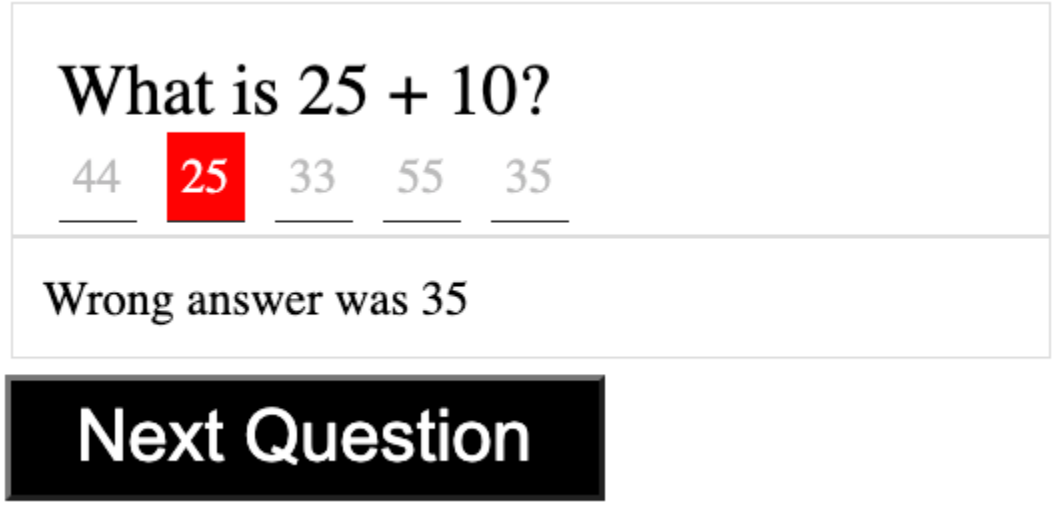
7. Create the doGet() function, using JSON.stringify() convert the returned results from the function with the Sheet data.
8. Return within the doGet() the string value of the object, outputting it into the web app contents. Set the MimeType as JSON.
9. Deploy the webapp as an open URL without permissions to access it. Check the web app URL in the browser to see the data.

```
function doGet(){
   const d = JSON.stringify(dataObj());
 return
ContentService.createTextOutput(d).setMimeType(ContentService.Mi
meType.JSON);
}

function dataObj(){
   const id = '1SEDiOufD8s********39pi0';
   const data =
SpreadsheetApp.openById(id).getSheetByName('questions').getDataR
ange().getValues();
   const headings = data[0];
   const rows = data.slice(1);
   const res = rows.map((row)=>{
     const temp = {question:row[0],answer:row[1],opts:[]};
     for(let i=1;i<6;i++){
       const val = row[i];
       if(val){
         temp.opts.push(val);
       }
     }
```

```
    return temp;
  })
  return res;
}
```

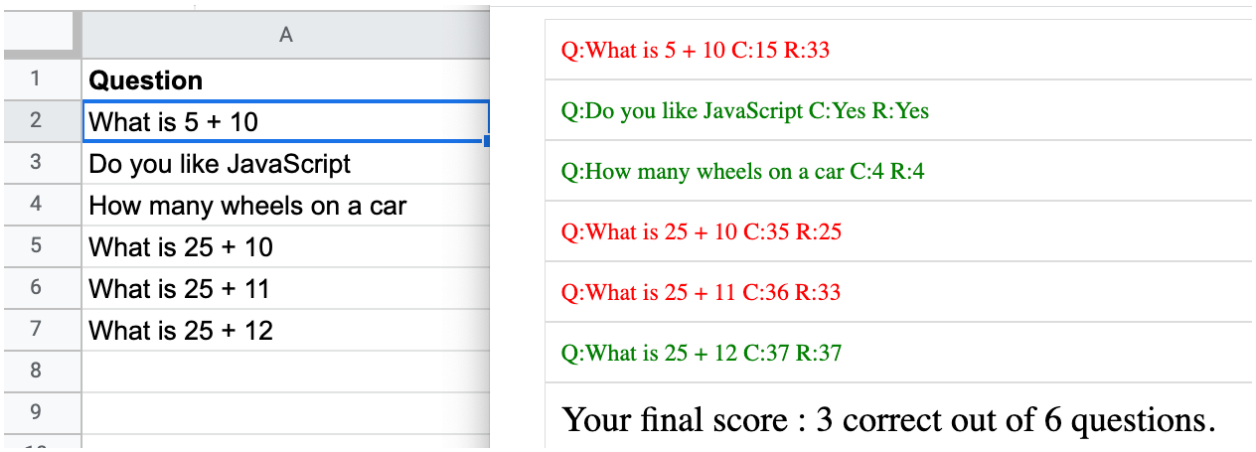
JavaScript Dynamic Quiz from Sheet Data

Creating an interactive application, from the JSON data that is loaded into the JavaScript code. The JSON data is coming live from a Google Spreadsheet.



Quiz application question screenshot

The quiz will load and dynamically generate the questions, total the score of the player and output the final player results.



Quiz final score results output

Exercise : Build an interactive dynamic Quiz game that uses Google Sheet data for the questions and answers.

1. Create HTML elements and CSS classes to be added to the dynamically created page elements within the JavaScript code
2. Select the page elements, create a main game object to track score and question data.
3. Add an event listener to the page button, when clicked it should start the game. Create a startGame() function.
4. Within the startGame() function using fetch load the Spreadsheet JSON data into the game JavaScript object. Then load the first question. Create a function to load and output the question. loadQuestion()
5. In the loadQuestion() function, get the game data first item, using the index value. Create a main element container, within the main element add a new div with the text content of the question value.
   Randomize the order of the array that holds the options for the question answers. Within the array sort method, you can add in a function that can return either a positive or negative value. This will change the sort and randomize in place the array contents. arr.sort(() => {return Math.random() - 0.5;})
6. Loop through the items in the options array, create elements with the option value. Add to the element object the values of the correct answer and the selected option value. Add an eventListener to listen for a click on the option.
7. Create a function that checks the correct answer, loop through all the options for the question and remove the event listener from them with removeEventListener('click', checker)
8. Check if the event target element, value of the correct answer matches the question answer from the Sheet data. If it does increment the score and provide feedback that the answer was correct. If it's incorrect, provide feedback that it is incorrect.
9. Create a new button. Check which question and how many questions are left. If there are no more questions, end the game. Add to the button the

endGame() function which will output the final score to the player. If there are still questions left then move to the next question and output the question contents for the player.

10. Once there are no more questions than end the game. Output to the player their results and score. You can add an array to the game object that tracks the players responses to each question to provide a final progress report.

HTML

```
<!doctype html>
<html>

<head>
  <title>JavaScript JSON AJAX</title>
  <style>
     .question {
        padding: 5px;
        font-size: 1.5em;
     }

     .main {
        border: 1px solid #ddd;
        padding: 10px;
        width: 90%;
        margin: auto;

     }

     .box {
        padding: 5px;
```

```
        border-bottom: 1px solid #333;
        margin: 5px;
    }

    .box1:hover {
        background-color: black;
        color: white;
        cursor: pointer;
    }

    .btn,
    .btn1 {
        display: block;
        margin: auto;
        width: 200px;
        margin: 5px;
        padding: 5px;
        color: white;
        background-color: black;
        font-size: 1.5em;
    }
  </style>
</head>

<body>
  <div>
    <div class="output"></div>
    <button class="btn">Click Me</button>
```

```
  </div>
  <script src="app1.js"></script>
</body>

</html>
```

JAVASCRIPT

```
const url =
'https://script.google.com/macros/s/AKfycbwjVbJStGI63rkZxd6UIuaD
nDx4Ocv5x5bhlji-7DSRpplsH5QpBx5QZta3m_AGZO4/exec';
const btn = document.querySelector('.btn');
const output = document.querySelector('.output');
btn.textContent = 'Start Game';
btn.onclick = startGame;
const game = {
   data: {},
   que: 0,
   score: 0,
   ans: []
};

function startGame() {
   btn.style.display = 'none';
   output.textContent = 'Starting game, loading data....';
   fetch(url)
       .then(res => res.json())
       .then(data => {
           game.data = data;
```

```
            buildGame();
        })
}

function buildGame() {
    console.log(game.data);
    loadQuestion();
}

function loadQuestion() {
    const q = game.data[game.que];
    output.innerHTML = '';
    const main = maker('div', output, 'main', '');
    const que = maker('div', main, 'question', `${q.question}?`);
    q.opts.sort(() => {
        return Math.random() - 0.5;
    });
    q.opts.forEach((el) => {
        const span = maker('span', main, 'box', el);
        span.correct = q.answer;
        span.selOpt = el;
        span.classList.add('box1');
        span.addEventListener('click', checker);
    })
    console.log(q);
}

function checker(e) {
```

```
    const el = e.target;
    game.ans.push(el.selOpt);
    const boxs = document.querySelectorAll('.box');
    boxs.forEach((ele) => {
        ele.removeEventListener('click', checker);
        ele.style.color = '#bbb';
        ele.classList.remove('box1');
        ele.disabled = true;
    })
    if (el.correct == el.selOpt) {
        game.score++;
        const main = maker('div', output, 'main', 'Correct
answer');
        el.style.color = 'white';
        el.style.backgroundColor = 'green';
    } else {
        const main = maker('div', output, 'main', `Wrong answer
was ${el.correct}`);
        el.style.color = 'white';
        el.style.backgroundColor = 'red';
    }
    const btn1 = maker('button', output, 'btn1', `Next
Question`);
    game.que++;
    const total = game.data.length - game.que;
    console.log(`remaining ${total}`);
    if (total == 0) {
        btn1.textContent = 'Game Over See Score';
```

```
        btn1.onclick = endGame;
    } else {
        btn1.onclick = loadQuestion;
    }
    console.log(el.correct, el.selOpt);
}

function endGame() {
    //output.textContent = JSON.stringify(game);
    output.innerHTML = '';
    game.ans.forEach((ele, ind) => {
        let html = `Q:${game.data[ind].question}
C:${game.data[ind].answer} R:${ele}`;
        const div = maker('div', output, 'main', html);
        const bg = (game.data[ind].answer == ele) ? 'green' :
'red';
        div.style.color = bg;
    })
    const htmlScore = `Your final score :  ${game.score} correct
out of ${game.data.length} questions.`;
    const div = maker('div', output, 'main', htmlScore);
    div.style.fontSize = '1.5em';
    console.log(game);
}


function maker(t, p, c, h) {
    const el = document.createElement(t);
```

```
    el.classList.add(c);
    el.innerHTML = h;
    return p.appendChild(el);
}
```

Submit content to Sheet POST

With Apps Script you can publish the script as a web app, and web apps can return content both with doGet(e) or doPost(e) functions. When a request sends the HTTP POST to the web app, the Apps Script runs doPost(e). The e argument represents an event parameter that can contain information about any request parameters. This can include the POST fetch body contents, which can then be viewed in the e parameter and e parameters like in the example below.

E.parameters send data with POST and body of score and name values in the formData object.
{"contextPath":"","parameter":{"score":"1000","name":"Laurence Svekis"},"contentLength":246,"parameters":{"score":["1000"],"name":["Laurence Svekis"]},"queryString":""}

The e.parameter is an object of key and value pairs that correspond to the request parameters. The parameters will return the key and values in an array, and can be used if the key has more than one value. For the parameter if more values are sent it only uses the first value.

In this exercise we will send data from the web page input fields, to the Google Apps Script WebApp URL as POST data. The data will then be inserted into the matching columns by the name of the column headings and the name of the object key.

Example of Spreadsheet columns and rows.

Name	Score
Laurence Smith	50
Laurence Jones	55
Laurence Svekis	1000

Jane	555

The HTML and JavaScript will create a simple interface with two input fields that match the column headings.

Name:
Laurence Svekis
Score:
1000
Click Me

Once the button is clicked, a fetch request will be sent to the endpoint from the Apps Script WebApp URL.

Data is inserted into the Spreadsheet sheet with the name of score.

	A	B
1	Name	Score
2	Laurence Smith	50
3	Laurence Jones	55
4	Laurence Svekis	1000

estions score

Exercise : Send a POST request to Google Apps Script WebApp URL and update a Spreadsheet

1. In Google Apps Script add the doPost() function.
2. Create an object in doPost() with a stringified the value of the e parameters, and will have a value for the id which is the row the data was inserted into.
3. Return the content as stringified values using ContentService and set a MimeType of JSON.
4. Create a function that is named updater() pass the e parameters to the updater() function. Using a separate function will allow for testing as otherwise to test doPost() you will need to redeploy each time.
5. In the updater() function get the sheet you want to use, and select the first row of values as an array.
6. Create a temporary array which can be used to append to the sheet row.
7. Loop through the values which are the sheet headings, using if(.. in ..) check for the heading value within the e parameter. If found then using the index value of the column heading, assign the value of the array to the value of that parameter value.
8. Append the temporary array of submitted values to the sheet row, return the sheet.getLastRow() which will be a numeric value of the last row of data or the row that the data was just inserted into.
9. Setup the HTML and create a JavaScript file.
10. Within the JavaScript code, select the page elements. Create two input fields one which will be the name the other which will be used as the score value.
11. Add an event click to the button, once clicked it should invoke a function that makes the fetch request.
12. Create a new formData object and append the name value and score value from the input fields.
13. Within the fetch set the options, method to POST and body to the formData.
14. Get the response back from the Google Apps Script as the row where the data was inserted.

HTML

```html
<!doctype html>
<html>

<head>
  <title>JavaScript JSON AJAX</title>
  <style>
     input{
        display:block;
     }
  </style>
</head>

<body>
  <div>
     <div class="output"></div>
     <button class="btn">Click Me</button>
  </div>
  <script src="app2.js"></script>
</body>

</html>
```

JAVASCRIPT

```javascript
const url = 'https://script.google.com/macros/s/AK***J1b9/exec';
const btn = document.querySelector('.btn');
const output = document.querySelector('.output');
const myInput1 = document.createElement('input');
const myInput2 = document.createElement('input');
```

```
myInput1.setAttribute('type','text');
myInput2.setAttribute('type','number');
myInput1.value = 'Laurence Svekis';
myInput2.value = 1000;
const label1 = document.createTextNode('Name:');
const label2 = document.createTextNode('Score:');
output.append(label1);
output.append(myInput1);
output.append(label2);
output.append(myInput2);
btn.onclick = sendData;

function sendData(){
   const formData = new FormData();
   formData.append('name',myInput1.value);
   formData.append('score',myInput2.value);
   fetch(url,{
       method : 'POST',
       body : formData
   }).then(rep => rep.json())
   .then(data =>{
       console.log(data);
   })
}
```

GOOGLE APPS SCRIPT

```
function updater(e){
```

```
 //const e =
{"contextPath":"","parameter":{"score":"50","name":"Laurence
Smith"},"contentLength":246,"parameters":{"score":["1000"],"name
":["Laurence Svekis"]},"queryString":""};
 const id = '1SEiO**9pi0';
 const sheet =
SpreadsheetApp.openById(id).getSheetByName('score');
 const headings = sheet.getDataRange().getValues()[0];
 Logger.log(headings);
 const arr = [];
 headings.forEach((heading,ind)=>{
   const val = heading.toLowerCase();
   Logger.log(val);
   if(val in e.parameter){
     Logger.log(e.parameter[val]);
     arr[ind] = e.parameter[val];
   }else{
     Logger.log(`${val} not found`);
   }
 })
 sheet.appendRow(arr);
 return sheet.getLastRow();
}

function doPost(e){
 const obj1 = {id:updater(e),json:JSON.stringify(e)};
 const output = JSON.stringify(obj1);
```

```
 return
ContentService.createTextOutput(output).setMimeType(ContentServi
ce.MimeType.JSON);
}
```

Complete Quiz with Score Submission to Google Sheets

Using the quiz code and the score submission for the previous lessons, this lesson will finalize the building of the interactive dynamic quiz application adding the ability for the user to submit their score into the Spreadsheet.

Once the quiz is complete, the user will be provided an input field asking for their name, and a button to submit the score. The score and number of questions which are in the global game object can be used to pass those values to the Google Apps Script endpoint URL with a POST request.

The Google Apps Script Web App can use both doGet() and doPost() and depending on the method, that will invoke the corresponding function and run its code. This way we can use just the one endpoint for both retrieving the question list as JSON, and sending the score details.

The quiz code will remain the same, the end of game with results should have the new input field added like in the example below.

Q:What is 5 + 10 C:15 R:15

Q:Do you like JavaScript C:Yes R:Yes

Q:How many wheels on a car C:4 R:4

Q:What is 25 + 10 C:35 R:25

Q:What is 25 + 11 C:36 R:36

Q:What is 25 + 12 C:37 R:37

Your final score : 5 correct out of 6 questions.

Name: Laurence Svekis

Send Score to Sheet

The doPost() from the Google Apps Script will automatically loop through all the sheet first row column values, so you can add a new value like total and then add to the formData on the submission the same property name and it will work. This is one of the benefits if the code is written to be dynamic as changes to the resources and updates to the data should be accommodated with little to no change in the final code.

Exercise : Create a way for users to submit their score at the end of the quiz.

1. Use the previous quiz code and copy in the post submission code.
2. Add to the end game screen, the input field and a button to submit the score.
3. Once the button is pressed invoke a function named sendeData(). Create the function with two parameters, the user name from the input field and the main element that contains both the input field and the button. This will allow the sendeData() to clear the contents of the element. Once the button

is pressed, update the main field to remove the button and the input and provide a message to the user.

4. In the sendeData() function, create a new formData object new FormData()
5. Append the desired value using the sheet column heading names as the key value, and the value that you want added into that column.
6. Using fetch, add the options to send the formData in the body and the method as POST
7. Once the response is received from the WebApp endpoint, you can provide feedback to the user. The response will contain the row which can then be used as an id confirmation that the score was submitted for the user if needed.

HTML

```
<!doctype html>
<html>
<head>
   <title>JavaScript JSON AJAX</title>
   <style>
       .question {
           padding: 5px;
           font-size: 1.5em;
       }
       .main {
           border: 1px solid #ddd;
           padding: 10px;
           width: 90%;
           margin: auto;
       }
       .box {
           padding: 5px;
```

```
            border-bottom: 1px solid #333;
            margin: 5px;
        }
        .box1:hover {
            background-color: black;
            color: white;
            cursor: pointer;
        }
        .btn,
        .btn1 {
            display: block;
            margin: auto;
            width: 200px;
            margin: 5px;
            padding: 5px;
            color: white;
            background-color: black;
            font-size: 1.5em;
        }
    </style>
</head>
<body>
    <div>
        <div class="output"></div>
        <button class="btn">Click Me</button>
    </div>
    <script src="quiz.js"></script>
</body>
```

```
</html>
```

JAVASCRIPT

```
const url =
'https://script.google.com/macros/s/AKfycbwKxmyrO0TOs8ONPO5eHNqq
AdrHaelft46HkTOklzuh5TFPwRh1DQqlkKDa-O-mJ1b9/exec';
const btn = document.querySelector('.btn');
const output = document.querySelector('.output');
btn.textContent = 'Start Game';
btn.onclick = startGame;
const game = {
   data: {},
   que: 0,
   score: 0,
   ans: []
};

function startGame() {
   btn.style.display = 'none';
   output.textContent = 'Starting game, loading data....';
   fetch(url)
       .then(res => res.json())
       .then(data => {
           game.data = data;
           buildGame();
       })
}
```

```
function buildGame() {
   loadQuestion();
}

function loadQuestion() {
   const q = game.data[game.que];
   output.innerHTML = '';
   const main = maker('div', output, 'main', '');
   const que = maker('div', main, 'question', `${q.question}?`);
   q.opts.sort(() => {
       return Math.random() - 0.5;
   });
   q.opts.forEach((el) => {
       const span = maker('span', main, 'box', el);
       span.correct = q.answer;
       span.selOpt = el;
       span.classList.add('box1');
       span.addEventListener('click', checker);
   })
}

function checker(e) {
   const el = e.target;
   game.ans.push(el.selOpt);
   const boxs = document.querySelectorAll('.box');
   boxs.forEach((ele) => {
       ele.removeEventListener('click', checker);
       ele.style.color = '#bbb';
```

```
        ele.classList.remove('box1');
        ele.disabled = true;
    })
    if (el.correct == el.selOpt) {
        game.score++;
        const main = maker('div', output, 'main', 'Correct
answer');
        el.style.color = 'white';
        el.style.backgroundColor = 'green';
    } else {
        const main = maker('div', output, 'main', `Wrong answer
was ${el.correct}`);
        el.style.color = 'white';
        el.style.backgroundColor = 'red';
    }
    const btn1 = maker('button', output, 'btn1', `Next
Question`);
    game.que++;
    const total = game.data.length - game.que;
    if (total == 0) {
        btn1.textContent = 'Game Over See Score';
        btn1.onclick = endGame;
    } else {
        btn1.onclick = loadQuestion;
    }
}

function endGame() {
```

```
    output.innerHTML = '';
    game.ans.forEach((ele, ind) => {
        let html = `Q:${game.data[ind].question}
C:${game.data[ind].answer} R:${ele}`;
        const div = maker('div', output, 'main', html);
        const bg = (game.data[ind].answer == ele) ? 'green' :
'red';
        div.style.color = bg;
    })
    const htmlScore = `Your final score :  ${game.score} correct
out of ${game.data.length} questions.`;
    const div = maker('div', output, 'main', htmlScore);
    div.style.fontSize = '1.5em';
    const userIn = maker('div', output, 'main', '');
    const myInput1 = document.createElement('input');
    myInput1.setAttribute('type', 'text');
    const label1 = document.createTextNode('Name:');
    userIn.append(label1);
    myInput1.value = '';
    userIn.append(myInput1);
    const btnSend = maker('button', userIn, 'btn1', 'Send Score
to Sheet');
    btnSend.onclick = () => {
        userIn.innerHTML = 'Score sent';
        sendeData(myInput1.value, userIn);
    }
}
```

```
function sendeData(valName, ele) {
   const formData = new FormData();
   formData.append('name', valName);
   formData.append('score', game.score);
   formData.append('total', game.data.length);
   fetch(url, {
           method: 'POST',
           body: formData
       }).then(rep => rep.json())
       .then(data => {
           ele.innerHTML = `Score Submitted Thank you`;
       })
}

function maker(t, p, c, h) {
   const el = document.createElement(t);
   el.classList.add(c);
   el.innerHTML = h;
   return p.appendChild(el);
}
```

GOOGLE APPS SCRIPT

```
function updater(e){
 const id = '1SE*******i0';
 const sheet =
SpreadsheetApp.openById(id).getSheetByName('score');
 const headings = sheet.getDataRange().getValues()[0];
 const arr = [];
```

```
 headings.forEach((heading,ind)=>{
   const val = heading.toLowerCase();
   if(val in e.parameter){
     arr[ind] = e.parameter[val];
   }})
 sheet.appendRow(arr);
 return sheet.getLastRow();
}

function doPost(e){
 const obj1 = {id:updater(e),json:JSON.stringify(e)};
 const output = JSON.stringify(obj1);
 return
ContentService.createTextOutput(output).setMimeType(ContentServi
ce.MimeType.JSON);
}

function doGet(){
   const d = JSON.stringify(dataObj());
 return
ContentService.createTextOutput(d).setMimeType(ContentService.Mi
meType.JSON);
}

function dataObj(){
   const id = '1SE*******39pi0';
```

```
  const data =
SpreadsheetApp.openById(id).getSheetByName('questions').getDataR
ange().getValues();
  const headings = data[0];
  const rows = data.slice(1);
  const res = rows.map((row)=>{
    const temp = {question:row[0],answer:row[1],opts:[]};
    for(let i=1;i<6;i++){
      const val = row[i];
      if(val){
        temp.opts.push(val);
      }
    }
    return temp;
  })
  return res;
}
```