Comma Separated Values or CSV is a type of text file where each value is delimited by a comma. CSV files are very useful for the import and export of data to other software applications.

Sometimes while developing web applications, you may come into a scenario where you need to download a CSV file converted from an HTML table. In this post, let us discuss how to create this feature using pure JavaScript without using any fancy plugins, modules, or frameworks.

**Approach:**

Create an HTML table. Then using JavaScipt and Document object module (DOM), we are going to extract each column data in a row and combine the data using commas. After doing this to each row, again using DOM, we are going to create a new download link and trigger the link using JavaScript event listeners to download the data so formed into a CSV file.

**Step-1: Create HTML table**

Create a simple HTML page with a table and a button. This button will be used as a trigger to convert the table into comma-separated values and download it in the form of a CSV file. Apply your own needed CSS stylings.

HTML

<html>

<body>

<center>

<h1 style="color:green" >GeeksForGeeks</h1>

<h2>Table to CSV converter</h2>

<table border="1" cellspacing="0" cellpadding="10">

<tr>

<th>Name</th>

<th>age</th>

<th>place</th>

</tr>

<tr>

<td>Laxman</td>

<td>19</td>

<td>Hyderabad</td>

</tr>

<tr>

<td>Dhoni</td>

<td>22</td>

<td>Ranchi</td>

</tr>

<tr>

<td>Kohli</td>

<td>25</td>

<td>Delhi</td>

</tr>

</table>

<br><br>

<button type="button" >download CSV</button>

</center>

</body>

</html>

**Step-2: Convert Table data into comma-separated values**

Write a Javascript function to retrieve the table data and convert it to comma-separated values. Make use of the document object module to access table data in each column of the rows. This function should be triggered when the user clicks the download button.

Javascript

function tableToCSV(){

// variable to store the final csv data

var csv\_data = [];

// get each row data

var rows = document.getElementsByTagName('tr');

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

// get each coloumn data

var cols = rows[i].querySelectorAll('td,th');

// stores each csv row data

var csvrow = [];

for(var j = 0; j < cols.length; j++){

// get the text data of each cell of a row and push it to csvrow

csvrow.push(cols[j].innerHTML);

}

// combine each coloumn value with comma

csv\_data.push(csvrow.join(","));

}

// combine each row data with new line character

csv\_data = csv\_data.join('\n');

/\*

we will use this function later to download the data in a csv file

//downloadCSVFile(csv\_data);

\*/

}

When the tableToCSV() function is triggered it access each table row data using the document object module. The getElementByTagName('tr') retrieves all table row data and stores it in rows variable. The rows[i].querySelectorAll('td,th') will get each column data of that table row. It is then stored in csvrow variable. The csvrow variable data are combined using commas to represent a row of a CSV file and then it is stored in a csv\_data variable which combinedly represents the data of our CSV file. We then join csv\_data values using the newline character as each row in a CSV file is represented in a new line. Now our data is ready to be exported into a CSV file.

**Step-3: Write a script to download the CSV file**

Now that we have our converted data ready, we need to write a script to create a CSV file, feed our data into it, and trigger the browser to download it automatically after the user has clicked the download button. Since this function will be triggered after the table data is converted, we will call this function inside tableToCSV() function.

Javascript

function downloadCSVFile(csv\_data){

// create CSV file object and feed our csv\_data into it

CSVFile = new Blob([csv\_data], {type: "text/csv"});

// create to temporary link to intiate dowload process

var temp\_link = document.createElement('a');

// download csv file

temp\_link.download = "GfG.csv";

var url = window.URL.createObjectURL(CSVFile);

temp\_link.href = url;

// this link should not be displayed

temp\_link.style.display = "none";

document.body.appendChild(temp\_link);

// automatically click the link to trigger dowload

temp\_link.click();

document.body.removeChild(temp\_link);

}

This function will take the CSV data that was formed earlier, as the argument. We will create a new file by creating a blob object of type CSV and then feed our CSV data into it. We need a link to trigger the browser window to download the file. However, we don't have any link in our HTML to do so. So, we will create a new link using DOM and provide its attributes with the appropriate values. This link so created should not be visible to the user as this link is solely for download triggering purposes and not for any other. So we need to make sure that this link is not visible to the user and is be removed once the download triggering process is over. Again we can use DOM to meet all our requirements. Using click() event listener, we can automatically let the link be clicked and download our CSV file. Now our CSV file should be successfully downloaded.

**Final Code:**

HTML

<html>

<body>

<center>

<h1 style="color:green" >GeeksForGeeks</h1>

<h2>Table to CSV converter</h2>

<table border="1" cellspacing="0" cellpadding="10">

<tr>

<th>Name</th>

<th>age</th>

<th>place</th>

</tr>

<tr>

<td>Laxman</td>

<td>19</td>

<td>Hyderabad</td>

</tr>

<tr>

<td>Dhoni</td>

<td>22</td>

<td>Ranchi</td>

</tr>

<tr>

<td>Kohli</td>

<td>25</td>

<td>Delhi</td>

</tr>

</table>

<br><br>

<button type="button" onclick="tableToCSV()" >download CSV</button>

</center>

</body>

<script type="text/javascript">

function tableToCSV(){

// variable to store the final csv data

var csv\_data = [];

// get each row data

var rows = document.getElementsByTagName('tr');

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

// get each coloumn data

var cols = rows[i].querySelectorAll('td,th');

// stores each csv row data

var csvrow = [];

for(var j = 0; j < cols.length; j++){

// get the text data of each cell of a row and push it to csvrow

csvrow.push(cols[j].innerHTML);

}

// combine each coloumn value with comma

csv\_data.push(csvrow.join(","));

}

// combine each row data with new line character

csv\_data = csv\_data.join('\n');

// call this function to download csv file

downloadCSVFile(csv\_data);

}

function downloadCSVFile(csv\_data){

// create CSV file object and feed our csv\_data into it

CSVFile = new Blob([csv\_data], {type: "text/csv"});

// create to temporary link to intiate dowload process

var temp\_link = document.createElement('a');

// download csv file

temp\_link.download = "GfG.csv";

var url = window.URL.createObjectURL(CSVFile);

temp\_link.href = url;

// this link should not be displayed

temp\_link.style.display = "none";

document.body.appendChild(temp\_link);

// automatically click the link to trigger dowload

temp\_link.click();

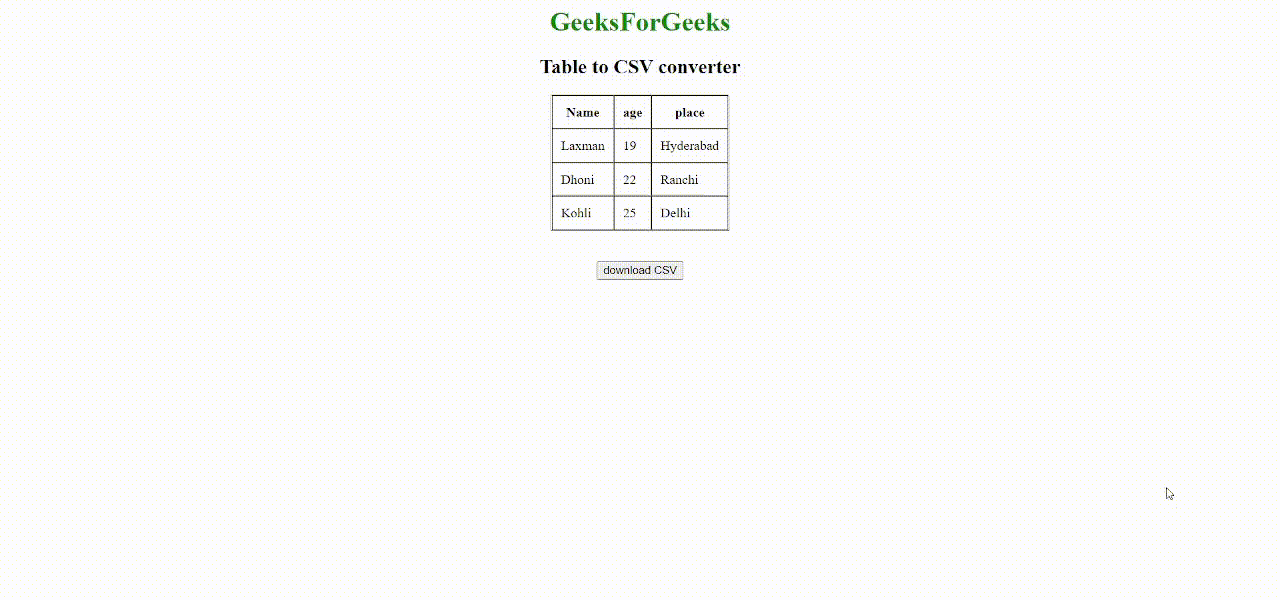
document.body.removeChild(temp\_link);

}

</script>

</html>

**Output:**

Html table to CSV file