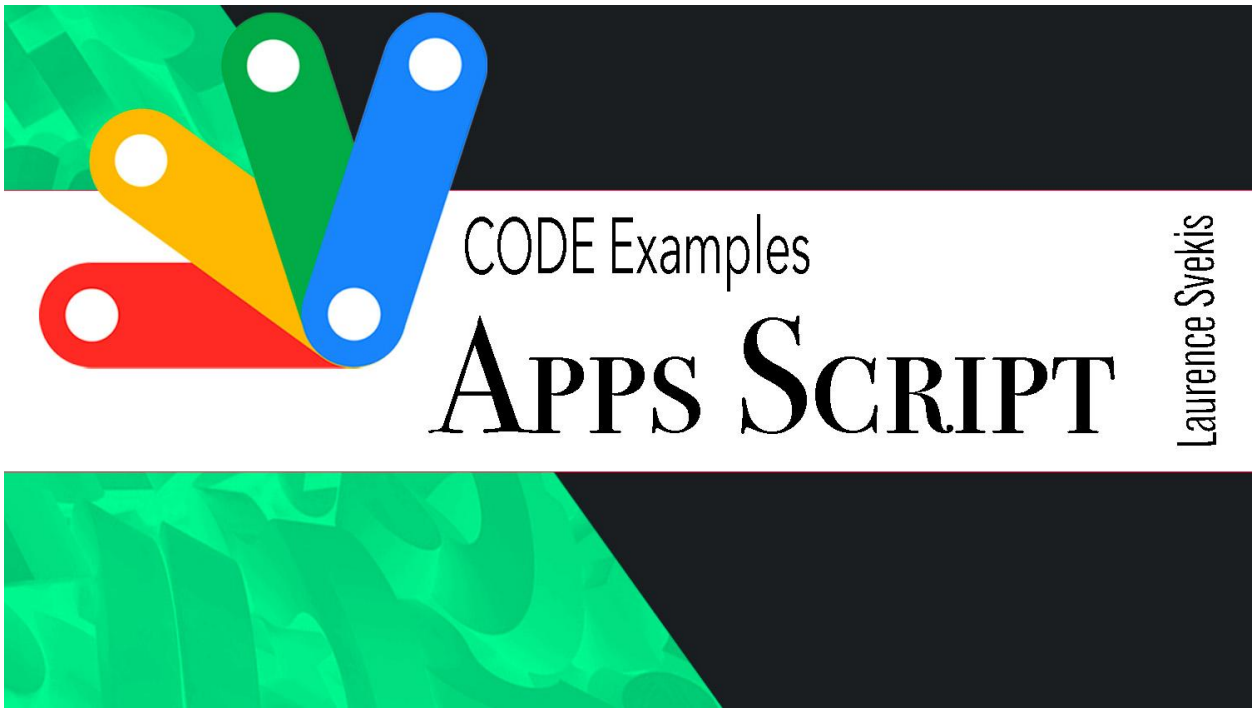
Apps Script Coding Examples V3



Create a Custom Function to Calculate Fibonacci Sequence:

https://youtu.be/OZipsYEcVJY

```
function fibonacci(n) {
  if (n <= 1) {
    return n;
  } else {
    return fibonacci(n-1) + fibonacci(n-2);
  }
}
```

Explanation: This script defines a custom function fibonacci that calculates the Fibonacci sequence up to a given number n. The function uses a recursive approach where the base case is when n is 0 or 1, and the recursive case is when n is greater than 1. The function returns the sum of the two previous numbers in the sequence to generate the next number.

E1:G1 | fx | =FIB(D1)

	A	B	C	D	E	F	G
1	3	4	5	6	8	21	10946

Custom function to calculate the Fibonacci sequence using Google Apps Script:

```
function fibonacci(n) {
  if (n <= 1) {
    return n;
  } else {
    return fibonacci(n-1) + fibonacci(n-2);
  }
}
```

To use this function in a Google Sheet, follow these steps:

Open a new or existing Google Sheet.

Click on the cell where you want to display the first number in the Fibonacci sequence.
Type the equal sign followed by the name of the function, "fibonacci".
Inside the parentheses, enter the number of the sequence you want to calculate.
For example, to calculate the 7th number in the Fibonacci sequence, type "=fibonacci(7)" in a cell.

The function uses a recursive approach to calculate the Fibonacci sequence. If the input number is less than or equal to 1, the function returns the input number as is. Otherwise, the function calls itself recursively with the input number decreased by 1 and 2, and returns the sum of the results. This approach generates the Fibonacci sequence by summing the two previous numbers in the sequence to generate the next number.

Note that custom functions in Google Sheets have some limitations, such as not being able to modify cells outside of the function's parent range and not supporting loops or prompts.

Send Emails to a List of Recipients from Google Sheets:

```
function sendEmails() {
  var sheet = SpreadsheetApp.getActiveSheet();
  var startRow = 2;  // First row of data to process
  var numRows = 3;   // Number of rows to process
  var dataRange = sheet.getRange(startRow, 1, numRows,
3);
```

```
  var data = dataRange.getValues();
  for (var i = 0; i < data.length; ++i) {
    var row = data[i];
    var emailAddress = row[0];  // First column
    var subject = row[1];       // Second column
    var message = row[2];       // Third column
    MailApp.sendEmail(emailAddress, subject, message);
  }
}
```

Explanation: This script retrieves data from a Google Sheet starting from row 2 and processes 3 rows. The data is retrieved using the getRange method and the getValues method retrieves the data from the range as a 2D array. The script then loops through each row in the array and retrieves the email address, subject, and message. Finally, the script sends an email to the recipient using the MailApp.sendEmail method.

Create a Custom Menu in Google Sheets:

https://youtu.be/uBVQFDdmXj4

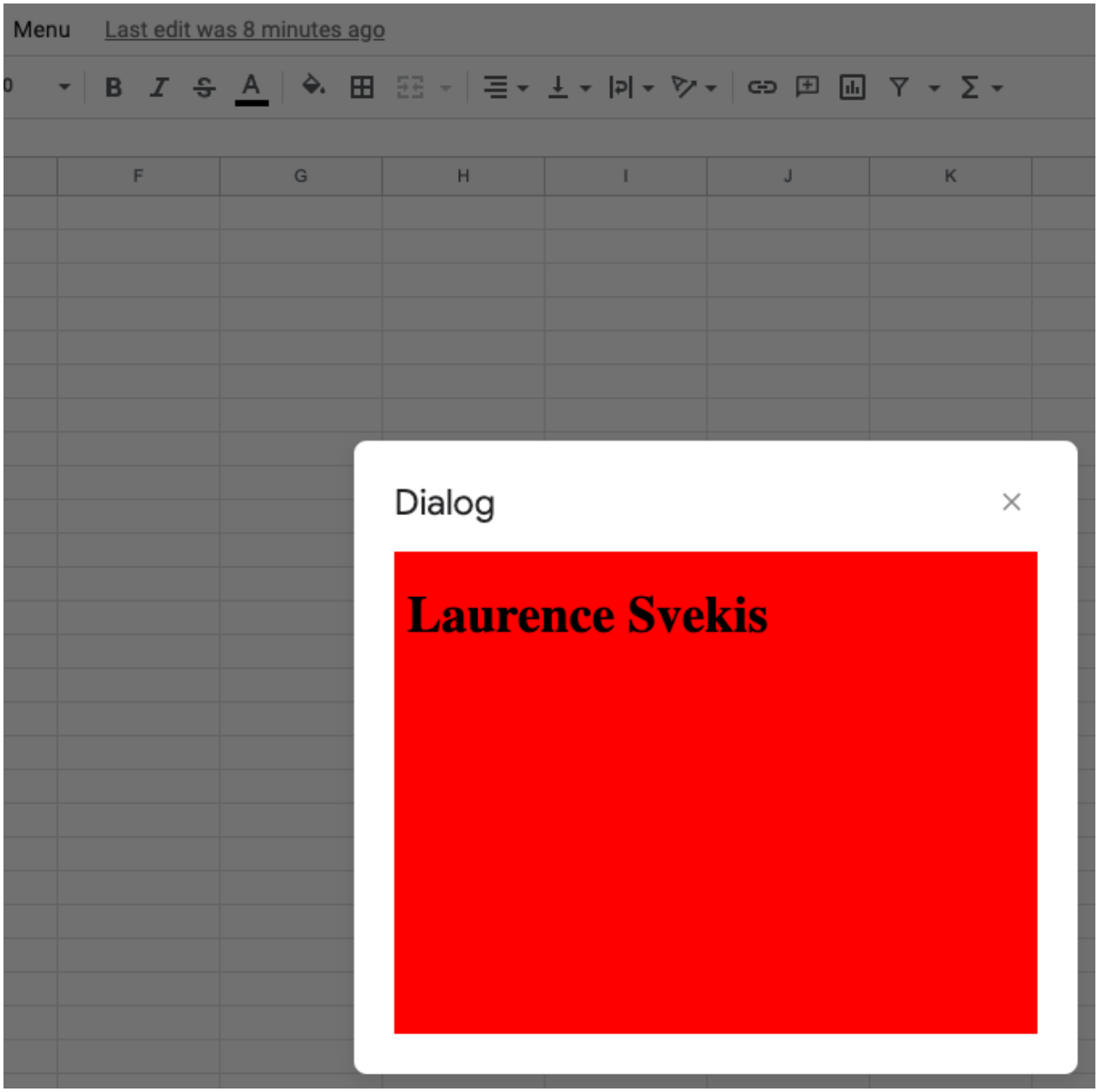
```
function onOpen() {
  var ui = SpreadsheetApp.getUi();
  ui.createMenu('Custom Menu')
      .addItem('Open Dialog', 'openDialog')
      .addToUi();
}

function openDialog() {
```

```
  var html =
HtmlService.createHtmlOutputFromFile('Page')
      .setWidth(400)
      .setHeight(300);
  SpreadsheetApp.getUi().showModalDialog(html, 'Dialog
Title');
}
```

Explanation: This script creates a custom menu in a Google Sheet using the onOpen trigger. The menu is called Custom Menu and it contains one item called Open Dialog. When the user clicks on this item, it calls the openDialog function, which creates an HTML dialog box using the HtmlService.createHtmlOutputFromFile method. The dialog box is displayed using the showModalDialog method and the title is set to Dialog Title.

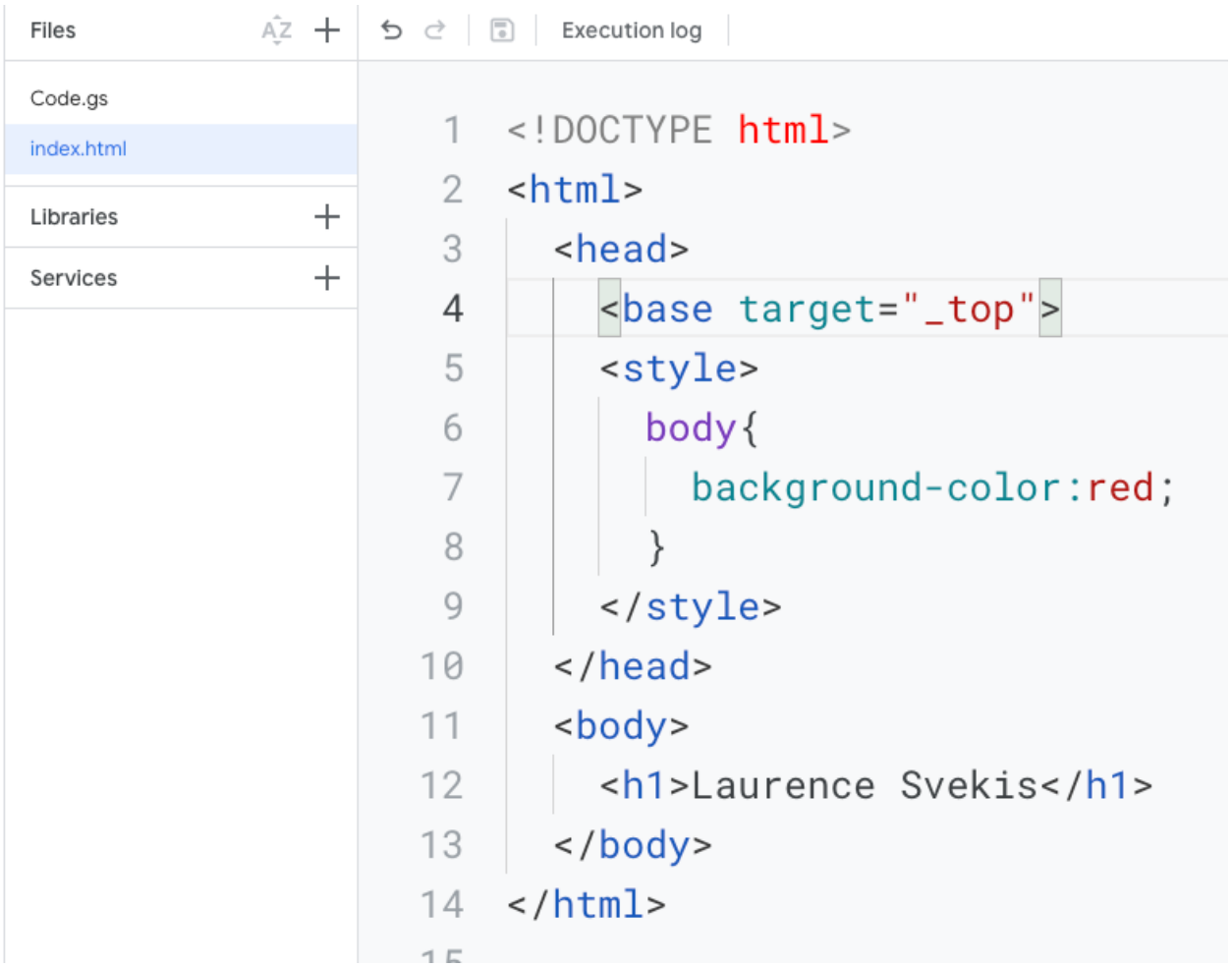
Dialog with HTML page contents



index.html file code

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
    <style>
```

```
      body{
        background-color:red;
      }
    </style>
  </head>
  <body>
    <h1>Laurence Svekis</h1>
  </body>
</html>
```




```
function onOpen(){
  const ui = SpreadsheetApp.getUi();
```

```
  ui.createMenu('Menu')
  .addItem('open','openDialog')
  .addToUi();
}

function openDialog(){
  const html =
HtmlService.createHtmlOutputFromFile('index')
  .setWidth(400)
  .setHeight(300);

SpreadsheetApp.getUi().showModalDialog(html,'Dialog');
}
```

Automatically Insert the Date in Google Sheets:

```
function onEdit(e) {
  var range = e.range;
  var column = range.getColumn();
  if (column == 1) {
    var row = range.getRow();
    var dateCell = range.offset(0, 1);
    dateCell.setValue(new Date());
  }
}
```

Explanation: This script automatically inserts the current date in a Google Sheet whenever a user edits a cell in the first column. The script uses the onEdit trigger to detect when a cell is edited and

retrieves the range of the edited cell. The script checks if the edited cell is in the first column using the getColumn method. If it is, the script retrieves the row and offsets the range by one column to the right using the offset method. Finally, the script sets the value of the offset cell to the current date using the setValue method.

Custom Log and onEdit Function

https://youtu.be/q-K6pS5BUIU

D12 fx

	A	B
1	Laurence Svekis 5	2/26/2023
2	Laurence Svekis	2/26/2023

```
function onEdit(e){
  const range = e.range;
  const col = range.getColumn();
  if(col == 1){
    const row = range.getRow();
    const dateCell = range.offset(0,1);
    dateCell.setValue(new Date());
    logN(col);
    logN(JSON.stringify(e));
  }
}

function logN(val){
```

```
  const ss =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName('l
og');
  ss.appendRow([val]);
}
```

Create a Custom Function to Capitalize the First Letter of Each Word in a String:

```
function capitalizeWords(str) {
  var words = str.split(' ');
  for (var i = 0; i < words.length; i++) {
    var word = words[i];
    var firstLetter = word.charAt(0).toUpperCase();
    var restOfWord = word.slice(1).toLowerCase();
    words[i] = firstLetter + restOfWord;
  }
  return words.join(' ');
}
```

Explanation: This script defines a custom function called capitalizeWords that capitalizes the first letter of each word in a string. The function splits the string into an array of words using the split method and then loops through each word. For each word, the function retrieves the first letter using the charAt method and capitalizes it using the toUpperCase method. The function also retrieves the rest of the word using the slice method and converts it to lowercase using the toLowerCase method. Finally, the function combines the first letter and the rest of the

word using the + operator and sets the value of the current word in the array. The function then returns the modified array of words joined back into a string using the join method.