Automatic Webscraping Guide for ChainExposed and implementation into Google Sheets

https://chainexposed.com/ ONLY

Implementing this script into your sheets appscript will get the data from the chainexposed link(s) you provide and feed them into your Sheet.

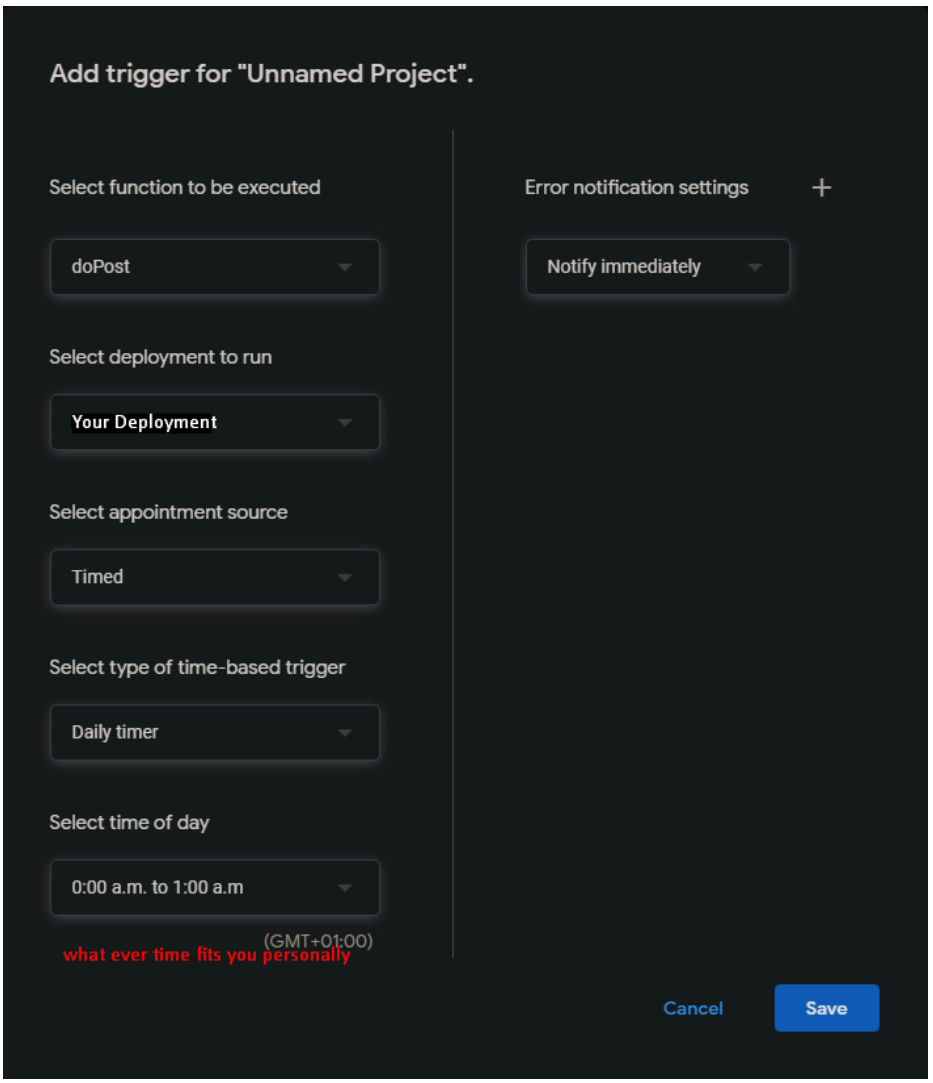
Tip: the script can theoretically handle an infinite amount of indicators - however the execution time limit is 6min. I tested it with 25 unique traces and came shy of 3min runtime

Guide:

1. copy code to your appscript
2. replace `SPREADSHEET` and `SHEET` with your personal id and name
3. in the urls() function add your link(s) like in the example
   (make sure to only run indicators with identical x-achsis in the same script to prevent errors
   - works together: dates (even with different starting dates)
   - doesn't work together: custom values that are different between indicators
4. when working with date based indicators I suggest putting the one with the most data in first
5. automatic data update - V1:

5.1. deploy as webApp

5.1. in appscript: left sidebar: "trigger" → bottom right: "add trigger" → copy the following settings:



6. automatic data update - V2:

6.1. deploy as webApp and copy url

6.2. set an alert in tv to trigger on your disiered timeframe and send to your webApp
(if you have problems see:
https://docs.google.com/document/d/1BWw9ZSdMHzhHfKD05EDOZLx9xZU4in-47Ra0ktiSaLU/edit?usp=sharing)

7. BEFORE you run your code: EXTEND YOUR SHEET TO AT LEAST 5000 ROWS
(otherwise your script will take forever to run and probably fail halfway through)

```
var URLs = urls();
let SPREADSHEET = '<your spreadsheet ID>';
let SHEET = '<your sheet name>';
var INDEX = 1;


function urls() {
```

```
  var urls = []
  urls.push('https://chainexposed.com/<link you want to add>');
  // add as many chainexposed links as you like - remember time limits

  return urls;
}

function doPost() {
  for (let i = 0; i < URLs.length; i++) scrape(URLs[i], i);
}

function scrape(url, urlIndex) {
  var response = UrlFetchApp.fetch(url);
  var htmlContent = response.getContentText();
  var script = extractScript(htmlContent);

  var traces = extractTraces(script);
  var data = []
  for (let i = 0; i < traces.length; i++) {
    var traceJson = extractJSON(traces[i]);
    if (traceJson === null) continue;
    if (i > 0 || urlIndex == 0) data.push(traceJson);
  }

  for (let i = 0; i < data.length; i++) {
    plotOnSheets(data[i], urlIndex);
  }
}



function extractScript(content) {
  var startIndex = content.indexOf('<main');
  var endIndex = content.indexOf('</main>', startIndex);
  let main = content.substring(startIndex, endIndex);

  let regex = /<script>[\s\S]*?<\/script>/g;
  var match = main.match(regex);
  if (match === null) return "";
  else return match[0];
```

```
}

function extractTraces(content) {
  let regex = /var\s*trace[0-9]*\s*=\s*{[\s\S]*?};/g;
  var traces = content.match(regex);

  if (traces !== null) return traces;
  else return '';
}

function extractJSON(content) {
  if (content.length < 100) return null;
  content = content.replace(/\s+/g, '');  // delete whitespaces
  content = content.replace(/'/g, '"');   // replace ' with "
  content = content.replace(/^var|;$/g, '');  // delete text in front
  content = content.replace(/^trace\d+=/, '');
  content = content.replace(/(\w+):/g, '"$1":');  // replace '<attribute>'
with "<attribute>"
  content = content.substring(0, content.length - 2) + "}";

  return JSON.parse(content);
}

function indexToColumn(index) {
  let column = '';
  while (index > 0) {
    let remainder = (index - 1) % 26;
    column = String.fromCharCode(65 + remainder) + column;
    index = Math.floor((index - 1) / 26);
  }
  return column;
}

function plotOnSheets(data, urlIndex) {
  let spreadsheet = SpreadsheetApp.openById(SPREADSHEET);
  let sheet = spreadsheet.getSheetByName(SHEET);

  if (urlIndex > 0) {
    let column = indexToColumn(INDEX);
```

```
    let start = sheet.getRange("A:A").getDisplayValues().map(function(row)
{return row[0]}).indexOf(data.x[0]);
    var j = 1; // compensate for different index start
    for (let i = 0; i < data.x.length; i++) {
      let field = column + (start + j);
      sheet.getRange(field).setValue(data.y[i]);
      j++;
    }
    INDEX++;
  }
  else {
    let spacing = 3  // row number where data from where data will be
placed
    if (INDEX > 1) {
      let column = indexToColumn(INDEX);
      for (let i = 0; i < data.x.length; i++) {
        let field = column + (i + spacing)
        sheet.getRange(field).setValue(data.y[i]);
      }
      INDEX++;
    }
    else {
      for (let i = 0; i < data.x.length; i++) {
        sheet.getRange('A' + (i + spacing)).setValue(data.x[i]); // date
        sheet.getRange('B' + (i + spacing)).setValue(data.y[i]); // price
      }
      INDEX += 2;
    }
  }
}
```

the data will appear in google sheets like this:

	A	B	C	D	E	F	G	H
1	2024-02-20	52308.05317						
2	Date 0	Price	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
1541	2016-03-18	408.9015874	398.6912778			0.3367684333	0.248118431	1.329996693
1542	2016-03-19	409.1244299	398.7156497			0.3372502414	0.2485769325	1.330808227
1543	2016-03-20	412.0569538	399.0908499			0.341873318	0.2538351394	1.340186402
1544	2016-03-21	411.6017741	399.4629896			0.341018125	0.2525804086	1.337936564
1545	2016-03-22	416.5816538	400.4230492			0.3475906039	0.260935321	1.35306155
1546	2016-03-23	417.3782995	400.7409571			0.3477419232	0.2621117165	1.3552187
1547	2016-03-24	413.5602399	400.9074298			0.339092477	0.2551395774	1.342533406
1548	2016-03-25	416.3732438	401.3876045			0.3429517145	0.2595444753	1.350520006
1549	2016-03-26	416.8230967	401.7795363			0.343333164	0.2599710242	1.351298439
1550	2016-03-27	425.4549598	402.8348456			0.3565756193	0.2743552696	1.378084837
1551	2016-03-28	422.9174468	403.444601			0.3522263483	0.2695173175	1.368957847
1552	2016-03-29	415.8638288	403.5469287			0.341023563	0.2570517809	1.345988824
1553	2016-03-30	412.7776978	403.610997	8.207484399	8.87733583	0.3361594951	0.2513774973	1.335786724
1554	2016-03-31	415.9568112	404.3124879	8.207484399	8.87733583	0.3413888496	0.256737556	1.345419788
1555	2016-04-01	417.2006603	404.5438559	8.475667462	8.579095251	0.3435745867	0.2589534114	1.349442822
1556	2016-04-02	419.7630567	404.8067559	8.301341131	8.591004858	0.3474890339	0.2631934398	1.357208328
1557	2016-04-03	420.0735789	405.3556209	7.563945236	8.273615135	0.3479022693	0.2635033293	1.35777939
1558	2016-04-04	419.5407279	405.8637852	7.976477769	8.156405536	0.3469914129	0.2623789445	1.355709673
1559	2016-04-05	422.5013644	406.3214845	7.702803321	8.063725938	0.3511849159	0.2671547561	1.364544572
1560	2016-04-06	421.9071912	406.9111993	7.872728556	8.014349696	0.3502135683	0.265430032	1.361340708
1561	2016-04-07	421.46722	407.7109987	8.978088936	8.124436059	0.3491759279	0.2630131035	1.356876228
1562	2016-04-08	418.6434605	407.8442242	8.209960701	8.08647795	0.3448726902	0.2580338207	1.347770327
1563	2016-04-09	418.7586354	408.1876973	8.288101027	8.084586507	0.3450401359	0.2581228606	1.347932086
1564	2016-04-10	421.9544885	408.4762139	7.792173163	8.117190496	0.349832611	0.2627654267	1.35642038
1565	2016-04-11	423.273044	408.6663054	6.812284977	7.95087724	0.3518175399	0.2647610859	1.360102112
1566	2016-04-12	427.4276646	409.2490188	9.054612684	8.143992864	0.3580705511	0.2712312668	1.37217742
1567	2016-04-13	424.8725181	409.5571797	10.06340304	8.456946361	0.3542315298	0.266706924	1.363711226
1568	2016-04-14	425.856595	409.791148	8.340324112	8.365837101	0.3557141687	0.268183173	1.366462157
1569	2016-04-15	430.7244188	410.3052364	9.760892963	8.587398852	0.3629330446	0.2760460299	1.381303289

the data will be fed in starting in row 3 - if you want to change that look for the comment in the code (spacing)
data will look for the same start date in the first column to reduce space