DEVELOPER

S

Data matcher

Verify whether the email is legal

Supports checking whether the email meets the specified requirements in the text type column, and if it does, it will be marked as √ in the corresponding checkbox column

Find the id card number in a column

Support to find the id card number of "xxxxx-xxxxx-x" where x are all numbers in a column of data and display it

Find the phone number in a column

Support finding and displaying phone numbers in a specified format in a column of data, such as +1-650-555-1234

Previous
« Find and replace

Next
Verify whether the email is legal »

Development

Preparation

Pre-knowledge

Before proceeding with the task, we assume that you are already familiar with JavaScript. Of course, if you do not know enough about JavaScript, you can refer to Appendix: JavaScript Getting Started Guide, which briefly explains JavaScript syntax knowledge and lists some introductory tutorial sites.

In addition, this script will also use the Script API, using these APIs you can use JavaScript to manipulate tables:

- Space API Space API, you can get datasheets under the space and do follow-up operations
- Datasheet API It can obtain datasheet information, and support operations on views, fields, and records in the data table
- View API can get view information and records under the specified view
- Field API can get the field information in the datasheet
- Record API can get the information of each record
- UI APIs can rende interactive standardized UI components, such as text input box, field selector, etc., and also support rendering specified content (text, tables, etc.)

It doesn't matter if you are not familiar with the API, we will use them a lot in the follow-up practice, so you can learn it in practice.

Target

Next, we will use JavaScript to develop a search and replace widget in the script widget. This function is useful in many scenarios, such as batch replacement for a misspelling of a proper noun, and so on.

Before we start, we need to disassemble the specific work of this task. We mainly need to implement the following steps:

1. Requires user input for which data to find and replace
    - Value to find
    - Value to replace
    - In which field to find and replace
2. Extract all the data in the table, then find all the data you want to find and replace
    - [Loop](javascript#Loop statement) through each record in the table and extract the data of the specified field in the record
    - [Determine whether](javascript#Conditional statements) the data contains the lookup value, if so, you need to replace it with the replacement value, and save the replaced data as an array
    - Finally, replace the saved data one by one with the data to be replaced in the table
3. Output the final result
    - Replaced successfully
    - Replacement failed

Let's start writing this widget together!For more information on how the sample code works, please see here.

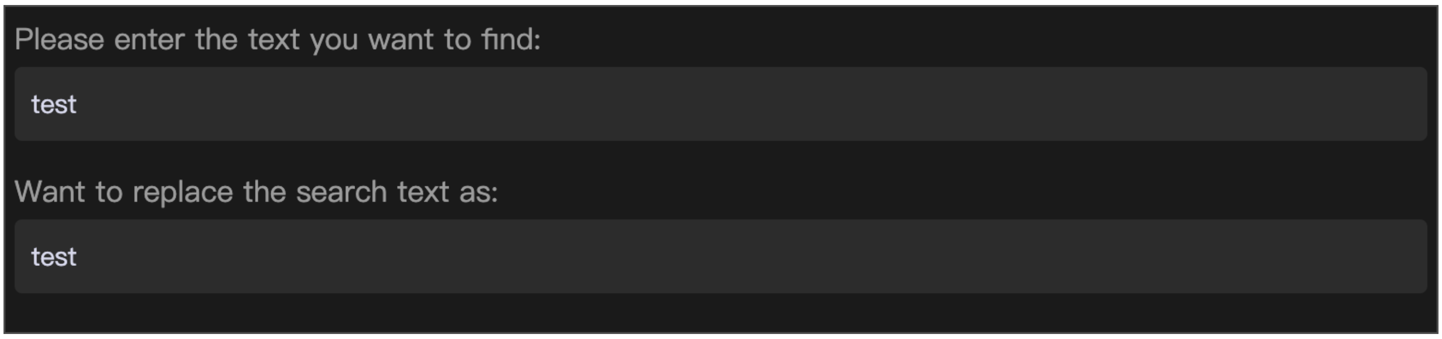
Begin to write

Use input API to input data

Input API can ask the user to enter the content, convenient for subsequent interaction.Therefore, we can use this API to ask the user to fill in the necessary information:

```
const findText = await input.textAsync(
  "Please enter the text you want to find:"
);
const replaceText = await input.textAsync(
  "Want to replace the search text as:"
);
```

The running effect is as follows:


Please enter the text you want to find:

test

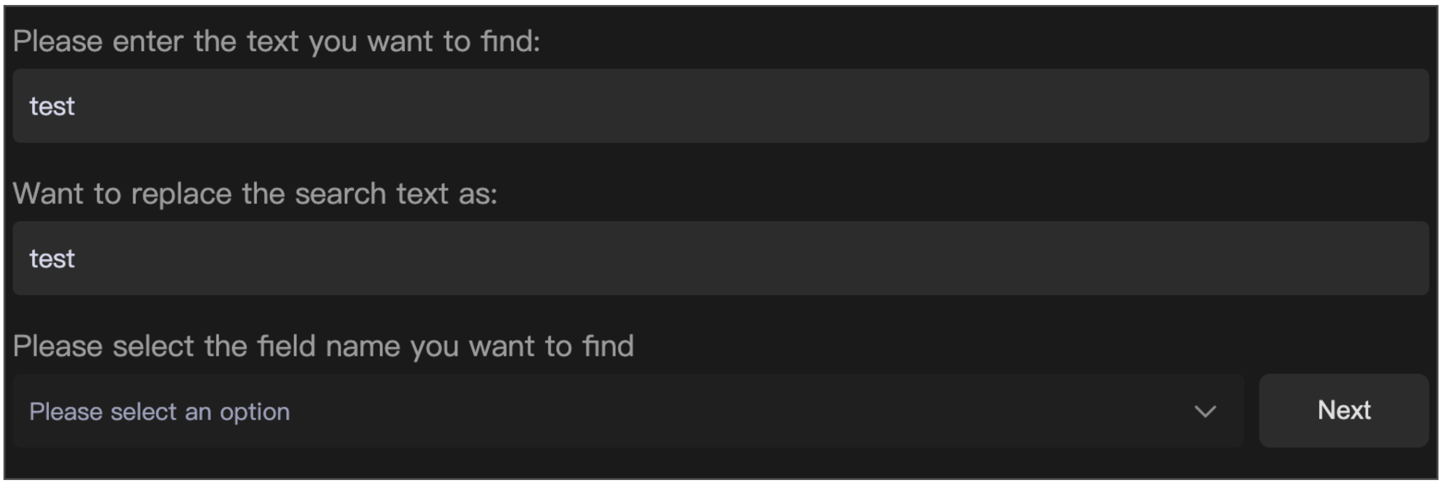
Want to replace the search text as:

test

After the above steps are done, we can get the text that users want to find replacement. Next, we need to let the user specify a field to facilitate us to find the data. Here we can use [Input.fieldasync]() to complete:

```
// You need to get the current datasheet through Space API and pass it as a parameter to input.fieldAsync
const datasheet = await space.getActiveDatasheetAsync();
const field = await input.fieldAsync(
  "Please select the field name you want to find",
  datasheet
);
```

The running effect is as follows:


Please enter the text you want to find:

test

Want to replace the search text as:

test

Please select the field name you want to find

Match data in the datasheet

After the above steps, we have completed the acquisition of all the necessary information. At this time, we should need to write the most important logical part.This part mainly uses JavaScript knowledge. First of all, we need to get the `id` of field we need to find. At this time, we can get it based on the `field` parameter obtained above, as shown below:

```
// Get the ID of filed
const fieldId = field.id;
```

Then use [datasheet.getRecordsAsync]() to get all the records in the current table, and create a new array to store the data that needs to be replaced:

```
// Get all record objects through the datasheet API
const records = await datasheet.getRecordsAsync();
const finalData = [];
```

At this time, the basic variable definition has been completed. The following needs to perform a loop traversal lookup for `records` and use [record.getCellValueString]() to find the `fieldId` data:

```
// traverse records
for (let record of records) {
  const recordId = record.id;
  // Get the data of the specified field in the record as cellValue
  const cellValue = record.getCellValueString(fieldId);
}
```

Then you need to use `<String>.replaceAll(findText, replaceText)` in JavaScript to find and replace data.This method can replace all `findText` in String with `replaceText`.However, since this method is only applicable to string types, it is necessary to determine whether the `cellValue` is empty before use. If it is empty, it means that there is no need to search and replace, and you can directly execute the next cycle with continue:

```
for (let record of records) {
  …… // If the acquired data is empty, jump out of this cycle and directly execute the next time
  if (cellValue == null) continue;

  // Replace findText in cellValue with replaceText, and use a new variable - newCellValue
  const newCellValue = cellValue.replace(findText, replaceText);
}
```

Finally, we need a [conditional statement](javascript#Conditional statements) to judge whether `newCellValue` is equal to `cellValue` `findText` If they are equal, it means that the data does not contain `findText` and no replacement is `cellValue` `newCellValue` to the previously defined `finalData`.The saved data format needs to refer to the updateRecordsAsync parameter in the Datasheet API:

```
for (let record of records) {
  …… // Determine whether the data before replacement is consistent with the data after replacement
  if (cellValue !== newCellValue) {
    // Inconsistency means that the corresponding record in the table needs to replace cellValue with newCellValue,
    finalData.push({
      id: recordId,
      valuesMap: { [fieldId]: newCellValue }
    })
  }
}
```

The final complete loop traversal search code is as follows:

```
// traverse records
for (let record of records) {
  const recordId = record.id;
  // Get the data of the specified field of the record as cellValue
  const cellValue = record.getCellValueString(fieldId);

  // If the acquired data is empty, jump out of this cycle and directly execute the next time
  if (cellValue == null) continue;

  // Replace findText in cellValue with replaceText, and use a new variable - newCellValue
  const newCellValue = cellValue.replaceAll(findText, replaceText);
  // Determine whether the data before replacement is consistent with the data after replacement
  if (cellValue !== newCellValue) {
    // Inconsistency means that the corresponding record in the table needs to replace cellValue with newCellValue,
    finalData.push({
      id: recordId,
      valuesMap: { [fieldId]: newCellValue },
    });
  }
}
```

At this point, we can already find all the data that needs to be replaced. At this point, we only need to use datasheet.updateRecordsAsync to replace the old data with the new data in `finalData`:

```
// Determine whether there is data in finalData, if there is no data, there is no need to replace
if (finalData.length) {
  await datasheet.updateRecordsAsync(finalData);
}
```

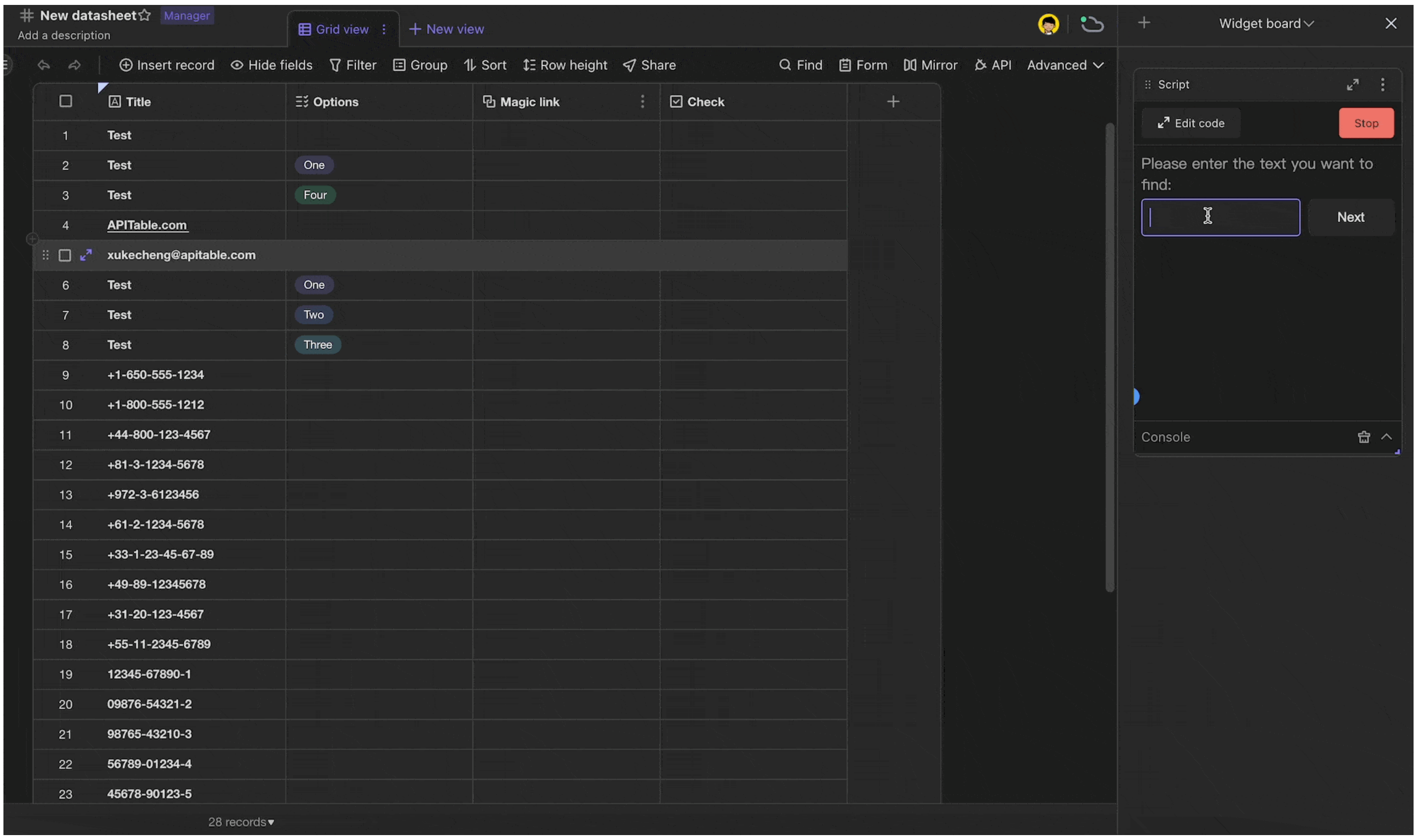
Output data using the Output API

In addition to supporting the Input API to allow users to input data, the script also supports the Output API to allow developers to output data and display it to users.

In the example of find and replace, we can add a reminder that the replacement is complete to the user to make the experience more complete:

```
// Determine whether there is data in finalData, if there is no data, there is no need to replace
if (finalData.length) {
  await datasheet.updateRecordsAsync(finalData);
  output.text("The replacement is complete!");
} else {
  output.text("No data found to be replaced");
}
```

Final effect and full code



Please enter the text you want to find:

	Title	Options	Magic link	Check
1	Test			
2	Test	One		
3	Test	Four		
4	APITable.com			
	xukecheng@apitable.com			
6	Test	One		
7	Test	Two		
8	Test	Three		
9	+1-650-555-1234			
10	+1-800-555-1212			
11	+44-800-123-4567			
12	+81-3-1234-5678			
13	+972-3-6123456			
14	+61-2-1234-5678			
15	+33-1-23-45-67-89			
16	+49-89-12345678			
17	+31-20-123-4567			
18	+55-11-2345-6789			
19	12345-67890-1			
20	09876-54321-2			
21	98765-43210-3			
22	56789-01234-4			
23	45678-90123-5			

```
const findText = await input.textAsync(
  "Please enter the text you want to find:"
);
const replaceText = await input.textAsync(
  "Want to replace the find text with:"
);

// The currently active form needs to be obtained through the Space API and passed to input.fieldAsync as a paramet
const datasheet = await space.getActiveDatasheetAsync();
const field = await input.fieldAsync(
  "Please select the field name you want to find",
  datasheet
);
// Get the id of field
const fieldId = field.id;

const records = await datasheet.getRecordsAsync();
const finalData = [];

// traverse records
for (let record of records) {
  const recordId = record.id;
  // Get the data of the specified field of the record as cellValue
  const cellValue = record.getCellValueString(fieldId);

  // If the acquired data is empty, jump out of this cycle and directly execute the next time
  if (cellValue == null) continue;

  // Replace findText in cellValue with replaceText, and use a new variable - newCellValue
  const newCellValue = cellValue.replaceAll(findText, replaceText);
  // Determine whether the data before replacement is consistent with the data after replacement
  if (cellValue !== newCellValue) {
    // Inconsistency means that the corresponding record in the table needs to replace cellValue with newCellValue,
    finalData.push({
      id: recordId,
      valuesMap: { [fieldId]: newCellValue },
    });
  }
}

// Determine whether there is data in finalData, if there is no data, there is no need to replace
if (finalData.length) {
  await datasheet.updateRecordsAsync(finalData);
  output.text("The replacement is complete!");
} else {
  output.text("No data found to be replaced");
}
```

Summary

Congratulations! you have implemented a "Find and Replace" widget with no more than 30 lines of code.

In addition, we have prepared more examples for you to further understand the purpose of script and more ways of writing.

Through this tutorial, I believe you have learned the usage of some APIs, and there are more APIs waiting for you to explore and discover more usages.

Previous
« How to install

Next
Code example »

DEVELOPER

S

Express inquiry

INFO

Support querying express delivery progress through express 100 API

placeholder, unfinished

```
const MD5 = function (e) {
  function h(a, b) {
    let c, d, e, f, g;
    e = a & 2147483648;
    f = b & 2147483648;
    c = a & 1073741824;
    d = b & 1073741824;
    g = (a & 1073741823) + (b & 1073741823);
    return c & d
      ? g ^ 2147483648 ^ e ^ f
      : c | d
      ? g & 1073741824
        ? g ^ 3221225472 ^ e ^ f
        : g ^ 1073741824 ^ e ^ f
      : g ^ e ^ f;
  }

  function k(a, b, c, d, e, f, g) {
    a = h(a, h(h((b & c) | (~b & d), e), g));
    return h((a << f) | (a >>> (32 - f)), b);
  }

  function l(a, b, c, d, e, f, g) {
    a = h(a, h(h((b & d) | (c & ~d), e), g));
    return h((a << f) | (a >>> (32 - f)), b);
  }

  function m(a, b, d, c, e, f, g) {
    a = h(a, h(h(b ^ d ^ c, e), g));
    return h((a << f) | (a >>> (32 - f)), b);
  }

  function n(a, b, d, c, e, f, g) {
    a = h(a, h(h(d ^ (b | ~c), e), g));
    return h((a << f) | (a >>> (32 - f)), b);
  }

  function p(a) {
    let b = "";
    let d = "";
    let c;
    for (c = 0; c <= 3; c++) {
      (d = (a >>> (8 * c)) & 255),
        (d = "0" + d.toString(16)),
        (b += d.substr(d.length - 2, 2));
    }
    return b;
  }

  let f = [];
  let q;
  let r;
  let s;
  let t;
  let a;
  let b;
  let c;
  let d;
  e = (function (a) {
    a = a.replace(/\r\n/g, "\n");
    for (var b = "", d = 0; d < a.length; d++) {
      const c = a.charCodeAt(d);
      c < 128
        ? (b += String.fromCharCode(c))
        : (c > 127 && c < 2048
            ? (b += String.fromCharCode((c >> 6) | 192))
            : ((b += String.fromCharCode((c >> 12) | 224)),
              (b += String.fromCharCode(((c >> 6) & 63) | 128))),
          (b += String.fromCharCode((c & 63) | 128)));
        (b += String.fromCharCode((c & 63) | 128)));
    }
    return b;
  })(e);
  f = (function (b) {
    let a;
    const c = b.length;
    a = c + 8;
    for (
      var d = 16 * ((a - (a % 64)) / 64 + 1), e = Array(d - 1), f = 0, g = 0;
      g < c;

    ) {
      (a = (g - (g % 4)) / 4),
        (f = (g % 4) * 8),
        (e[a] |= b.charCodeAt(g) << f),
        g++;
    }
    a = (g - (g % 4)) / 4;
    e[a] |= 128 << ((g % 4) * 8);
    e[d - 2] = c << 3;
    e[d - 1] = c >>> 29;
    return e;
  })(e);
  a = 1732584193;
  b = 4023233417;
  c = 2562383102;
  d = 271733878;
  for (e = 0; e < f.length; e += 16) {
    (q = a),
      (r = b),
      (s = c),
      (t = d),
      (a = k(a, b, c, d, f[e + 0], 7, 3614090360)),
      (d = k(d, a, b, c, f[e + 1], 12, 3905402710)),
      (c = k(c, d, a, b, f[e + 2], 17, 606105819)),
      (b = k(b, c, d, a, f[e + 3], 22, 3250441966)),
      (a = k(a, b, c, d, f[e + 4], 7, 4118548399)),
      (d = k(d, a, b, c, f[e + 5], 12, 1200080426)),
      (c = k(c, d, a, b, f[e + 6], 17, 2821735955)),
      (b = k(b, c, d, a, f[e + 7], 22, 4249261313)),
      (a = k(a, b, c, d, f[e + 8], 7, 1770035416)),
      (d = k(d, a, b, c, f[e + 9], 12, 2336552879)),
      (c = k(c, d, a, b, f[e + 10], 17, 4294925233)),
      (b = k(b, c, d, a, f[e + 11], 22, 2304563134)),
      (a = k(a, b, c, d, f[e + 12], 7, 1804603682)),
      (d = k(d, a, b, c, f[e + 13], 12, 4254626195)),
      (c = k(c, d, a, b, f[e + 14], 17, 2792965006)),
      (b = k(b, c, d, a, f[e + 15], 22, 1236535329)),
      (a = l(a, b, c, d, f[e + 1], 5, 4129170786)),
      (d = l(d, a, b, c, f[e + 6], 9, 3225465664)),
      (c = l(c, d, a, b, f[e + 11], 14, 643717713)),
      (b = l(b, c, d, a, f[e + 0], 20, 3921069994)),
      (a = l(a, b, c, d, f[e + 5], 5, 3593408605)),
      (d = l(d, a, b, c, f[e + 10], 9, 38016083)),
      (c = l(c, d, a, b, f[e + 15], 14, 3634488961)),
      (b = l(b, c, d, a, f[e + 4], 20, 3889429448)),
      (a = l(a, b, c, d, f[e + 9], 5, 568446438)),
      (d = l(d, a, b, c, f[e + 14], 9, 3275163606)),
      (c = l(c, d, a, b, f[e + 3], 14, 4107603335)),
      (b = l(b, c, d, a, f[e + 8], 20, 1163531501)),
      (a = l(a, b, c, d, f[e + 13], 5, 2850285829)),
      (d = l(d, a, b, c, f[e + 2], 9, 4243563512)),
      (c = l(c, d, a, b, f[e + 7], 14, 1735328473)),
      (b = l(b, c, d, a, f[e + 12], 20, 2368359562)),
      (a = m(a, b, c, d, f[e + 5], 4, 4294588738)),
      (d = m(d, a, b, c, f[e + 8], 11, 2272392833)),
      (c = m(c, d, a, b, f[e + 11], 16, 1839030562)),
      (b = m(b, c, d, a, f[e + 14], 23, 4259657740)),
      (a = m(a, b, c, d, f[e + 1], 4, 2763975236)),
      (d = m(d, a, b, c, f[e + 4], 11, 1272893353)),
      (c = m(c, d, a, b, f[e + 7], 16, 4139469664)),
      (b = m(b, c, d, a, f[e + 10], 23, 3200236656)),
      (a = m(a, b, c, d, f[e + 13], 4, 681279174)),
      (d = m(d, a, b, c, f[e + 0], 11, 3936430074)),
      (c = m(c, d, a, b, f[e + 3], 16, 3572445317)),
      (b = m(b, c, d, a, f[e + 6], 23, 76029189)),
      (a = m(a, b, c, d, f[e + 9], 4, 3654602809)),
      (d = m(d, a, b, c, f[e + 12], 11, 3873151461)),
      (c = m(c, d, a, b, f[e + 15], 16, 530742520)),
      (b = m(b, c, d, a, f[e + 2], 23, 3299628645)),
      (a = n(a, b, c, d, f[e + 0], 6, 4096336452)),
      (d = n(d, a, b, c, f[e + 7], 10, 1126891415)),
      (c = n(c, d, a, b, f[e + 14], 15, 2878612391)),
      (b = n(b, c, d, a, f[e + 5], 21, 4237533241)),
      (a = n(a, b, c, d, f[e + 12], 6, 1700485571)),
      (d = n(d, a, b, c, f[e + 3], 10, 2399980690)),
      (c = n(c, d, a, b, f[e + 10], 15, 4293915773)),
      (b = n(b, c, d, a, f[e + 1], 21, 2240044497)),
      (a = n(a, b, c, d, f[e + 8], 6, 1873313359)),
      (d = n(d, a, b, c, f[e + 15], 10, 4264355552)),
      (c = n(c, d, a, b, f[e + 6], 15, 2734768916)),
      (b = n(b, c, d, a, f[e + 13], 21, 1309151649)),
      (a = n(a, b, c, d, f[e + 4], 6, 4149444226)),
      (d = n(d, a, b, c, f[e + 11], 10, 3174756917)),
      (c = n(c, d, a, b, f[e + 2], 15, 718787259)),
      (b = n(b, c, d, a, f[e + 9], 21, 3951481745)),
      (a = h(a, q)),
      (b = h(b, r)),
      (c = h(c, s)),
      (d = h(d, t));
  }
  return (p(a) + p(b) + p(c) + p(d)).toLowerCase();
};

const intro = `
### 使用前需知
需要调用快递 100 API 对快递进行查询，因此需要你输入快递 100 企业后台（https://api.kuaidi100.com）中的授权参数才能使用
`;

output.markdown(intro);

// const key = await input.textAsync("请输入授权 key：");
// const customer = await input.textAsync("请输入 customer：");

const key = "RBMHPFRp8146";
const customer = "E24C3FCC31114560DE29E599E580DBB4";

const datasheet = await space.getActiveDatasheetAsync();
const numField = await input.fieldAsync("请选择快递单号所在列：", datasheet);
const numFieldId = numField.id;

const comField = await input.fieldAsync(
  "请选择快递公司编码所在列（示例数据 https://api.kuaidi100.com/manager/openapi/download/kdbm.do）：",
  datasheet
);
const comFieldId = comField.id;

const record = await input.recordAsync("请选择要查询的记录：", datasheet);
const recordId = record.id;

const num = record.getCellValueString(numFieldId);
const com = record.getCellValueString(comFieldId);

if (num != null && com != null) {
  console.log(record);
  const param = '{"com":"' + com + '","num":"' + num + '"}';
  const str = param + key + customer;
  console.log(str);
  const sign = MD5(str).toString().toUpperCase();
  console.log(sign);

  const data = {
    customer,
    param,
    sign,
  };

  let response = await fetch("https://poll.kuaidi100.com/poll/query.do", {
    method: "POST",
    headers: {
      "Content-Type": "application/x-www-form-urlencoded",
    },
    body: data,
  });

  const responseData = response.json();
  console.log(responseData);
  const responseMessage = responseData["message"];

  switch (responseMessage) {
    case "ok":
      const statusList = {
        0: "快件在途中",
        1: "快件揽件",
        2: "快件存在疑难",
        3: "快件已签收",
        4: "快件退签",
        5: "快件正在派件",
        6: "快件正处于返回发货人的途中",
        7: "快件转给其他快递公司邮寄",
        8: "快件清关",
        14: "收件人拒签快件",
      };
      const statusText = statusList[responseData["status"]];
      const str = `
      ### 快递当前状态
      ${statusText}
      `;
      output.markdown(str);

      const expressData = responseData["data"];

      const result = expressData.map((a) => {
        return { 时间: a.time, 详情: a.context };
      });
      output.table(result);
      break;
    default:
      output.clear();
      output.text(responseMessage);
      break;
  }
} else {
  output.clear();
  output.text("缺少快递单号或快递公司编码");
}
```

DEVELOPER

S

[Getting Started](#)[Examples](#)[Data matcher](#)Find the id card number in a column

On this page

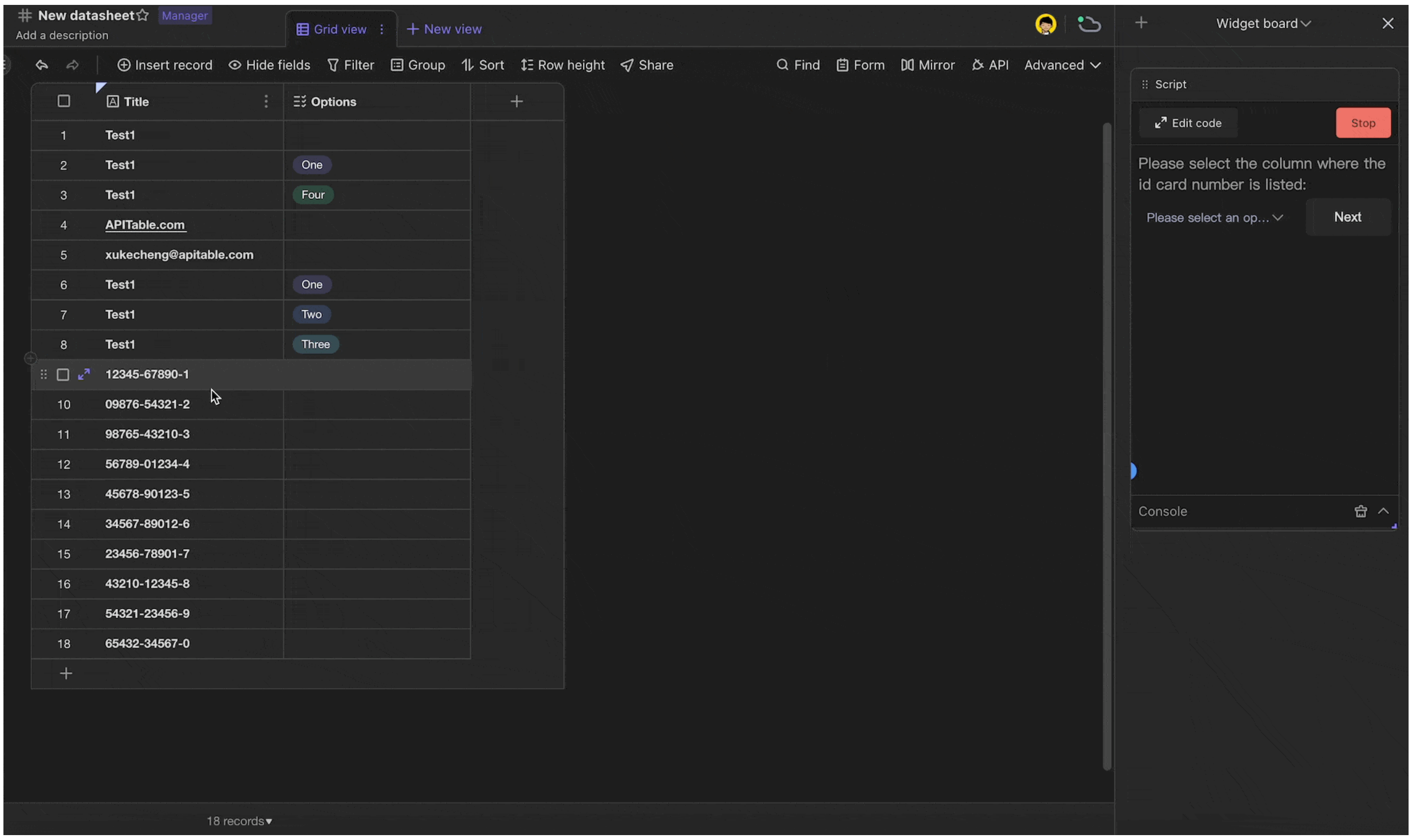
Find the id card number in a column

INFO

Support to find the id card number of "xxxxx-xxxxx-x" where x are all numbers in a column of data and display it

DEMO

The following is example data, not real ID Card number



	Title	Options
1	Test1	
2	Test1	One
3	Test1	Four
4	APITable.com	
5	xukecheng@apitable.com	
6	Test1	One
7	Test1	Two
8	Test1	Three
	12345-67890-1	
10	09876-54321-2	
11	98765-43210-3	
12	56789-01234-4	
13	45678-90123-5	
14	34567-89012-6	
15	23456-78901-7	
16	43210-12345-8	
17	54321-23456-9	
18	65432-34567-0	

Please select the column where the id card number is listed:

Source Code

```
const datasheet = await space.getActiveDatasheetAsync();
const idCardField = await input.fieldAsync("Please select the text column where the id card number is listed:", dat
const idCardFieldId = idCardField.id;

// you can customize this regex to suit your needs
const regexReg = /^\d{5}-\d{5}-\d$/;

const records = await datasheet.getRecordsAsync();
const finalData = ['Matched ID Card Number'];

for (let record of records) {
  let cellValue = record.getCellValue(idCardFieldId);

  if (cellValue == null) continue;

  const validation = cellValue.match(regexReg);

  if (validation != null) {
    finalData.push(record.getCellValueString(idCardFieldId));
  }
}

if (finalData.length){
  output.table(finalData);
} else {
  output.text("No matching data")
}
```

Previous
« Verify whether the email is legal

Next
Find the phone number in a column »

Developer Center

Development guide

API Reference

Social

Twitter

More

Homepage

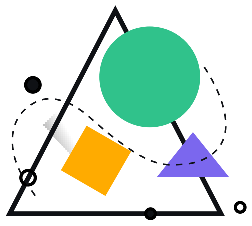
Help Center

GitHub

Open Source & AI-Oriented

Build your own AI ChatGPT / Copilot with your online editing datasets

Getting Started



REST API

Integration With AITable, Build Your Own Website Faster

De

De

AI

So

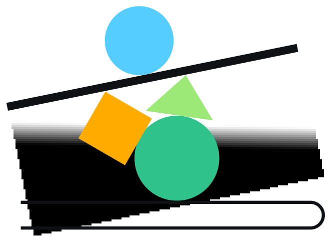
Tw

M

Hc

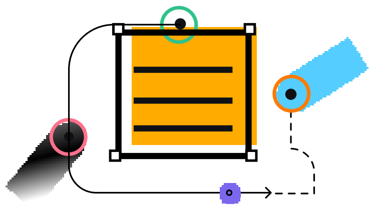
Help Center

Gi



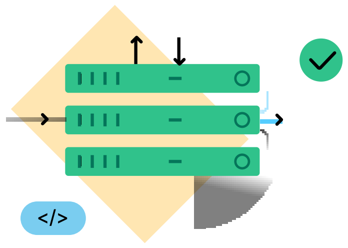
Widget SDK

Custom Widgets To Extend Table And View Capabilities



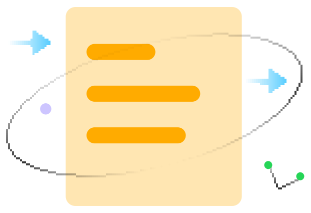
Script

Automated Workflows Deeply Integrated With AITable



Self-hosted

Learn About Deploying AITable On Your Own Infrastructure



Visual Database Technical White Paper

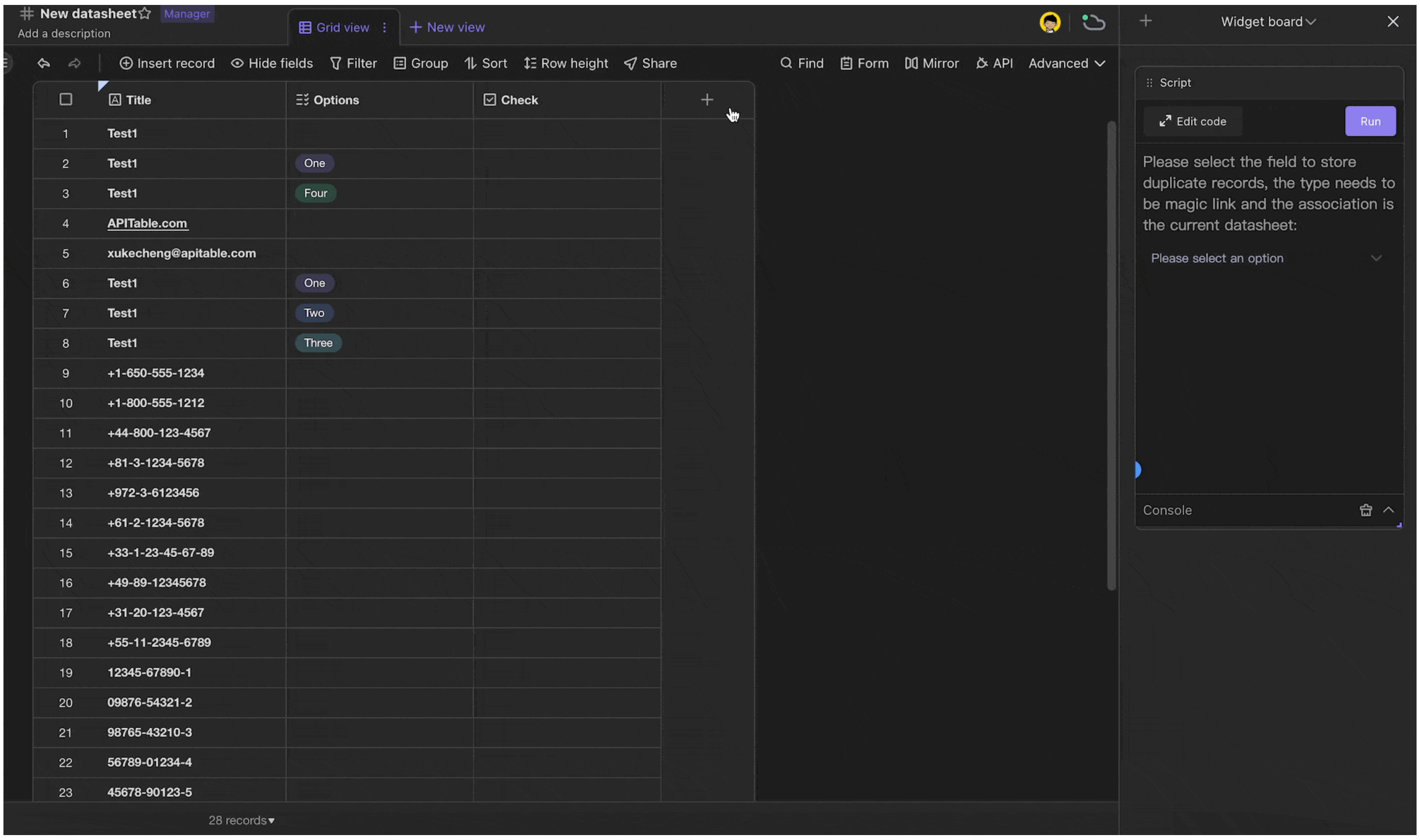
In-Depth Explanation Of The Internal Technical Mechanism Behind AITable's Spreadsheet-Database System

Link records with duplicate values

INFO

Select a record and the column to be checked, the script will automatically find the rest of the duplicate records and then associate them to the selected record (you need to create a Two-way link field in advance)

Demo



	Title	Options	Check
1	Test1		
2	Test1	One	
3	Test1	Four	
4	APITable.com		
5	xukecheng@apitable.com		
6	Test1	One	
7	Test1	Two	
8	Test1	Three	
9	+1-650-555-1234		
10	+1-800-555-1212		
11	+44-800-123-4567		
12	+81-3-1234-5678		
13	+972-3-6123456		
14	+61-2-1234-5678		
15	+33-1-23-45-67-89		
16	+49-89-12345678		
17	+31-20-123-4567		
18	+55-11-2345-6789		
19	12345-67890-1		
20	09876-54321-2		
21	98765-43210-3		
22	56789-01234-4		
23	45678-90123-5		

Please select the field to store duplicate records, the type needs to be magic link and the association is the current datasheet:

Please select an option

Source code

```
const datasheet = await space.getActiveDatasheetAsync();

const storeField = await input.fieldAsync(
  "Please select the field to store duplicate records, the type needs to be two-way link and the association is the
  datasheet
);
if (storeField.type != "TwoWayLink") {
  output.text(
    "Failed to run: The selected field type needs to be a Two-way link"
  );
} else if (storeField.property.foreignDatasheetId != datasheet.id) {
  output.text(
    "Failed to run: the selected magical link field must be linked to current datasheet"
  );
} else {
  console.log(storeField.property);

  const needFindRecord = await input.recordAsync(
    "Please select the record to check:",
    datasheet
  );
  const needFindField = await input.fieldAsync(
    "Please select the fields to check for duplicates:",
    datasheet
  );

  const records = await datasheet.getRecordsAsync();

  // Define two data sets that need to be used later, duplicates stores the duplicate record ids found, and valuesM
  const duplicates = [];
  const valuesMap = {};

  for (let record of records) {
    //The data that is empty does not belong to the content to be found, so it is skipped
    if (record.getCellValueString(needFindField.id) === null) {
      continue;
    }
    //Need to exclude yourself when checking for duplicates
    if (record.id === needFindRecord.id) {
      continue;
    }
    //Obtain data according to getCellValueString and judge whether they are equal. If they are equal, it means tha
    if (
      record.getCellValueString(needFindField.id) ===
      needFindRecord.getCellValueString(needFindField.id)
    ) {
      duplicates.push(record.id);
    }
  }

  console.log(duplicates);

  if (duplicates.length === 0) {
    output.text(`No duplicate records found`);
  } else {
    output.text(
      `Found ${duplicates.length} duplicate records, which have been linked in the Two-way link field`
    );
    valuesMap[storeField.id] = duplicates;
    await datasheet.updateRecordAsync(needFindRecord.id, valuesMap);
  }
}
```

Previous
« Find the phone number in a column

Next
Extract URL from attachment »

Homepage

Help Center

GitHub

Appendix: JavaScript Getting Started Guide

What is JavaScript?

JavaScript is a programming language and one of the most popular programming languages in the world.It has a profound impact on all aspects of our lives, usually open the browser to see all the web pages, a lot of App and so on can see it.

JavaScript itself is a very easy-to-use programming language, which is very suitable for beginner programming of rookies without any experience.At the same time, JavaScript is supported by most major browsers, and you can run JavaScript code without deploying any environment and only one browser.

Tutorial recommendation

JavaScript itself is a very easy-to-use programming language, which is very suitable for beginner programming of rookies without any experience.At the same time, JavaScript is supported by most major browsers, and you can run JavaScript code without deploying any environment and only one browser.

- MDN JavaScript Guide
- javascript.info
- javascript.info in Chinese

Getting started

For more information on how the sample code works, please see here.

Basic grammar

Like most programming languages, JavaScript code is made up of statements.For example, the following code snippet contains one statement per line, with a total of two lines:

```
console.log("Hello World");
const name = "bob";
```

Meanwhile, by default, each statement needs to end with `;`.In theory, we also support two statements on the same line, but this is not recommended:

```
console.log("Hello World");
const name = "bob";
```

Under normal circumstances, in order to achieve a function, many different statements will be written, and different statements also have different functions.It is the corresponding logic on the collocation of these different statements that constitute the code fragments in our daily cognition.The following is an overview of some of the basic writing methods and specific functions of JavaScript.

Data types and Variables

Just like the column types of AITable, different data types are defined in JavaScript.

String

One of the most frequently used data types is any text enclosed in single or double quotes.

```
const name = "bob";
```

Number

Numbers do not need to be enclosed in single or double quotation marks, so the wrapped numbers are not of the Number type.

```
const myString = "1"; // String
const myNumber = 1; // Number
```

Boolean

For Boolean values, there are only two cases: `true` and `false`, again without quotation marks.

```
const myString = "false"; // String
const myFalse = false; // Boolean
```

Array

Array is a little more complex than the above data types, it can contain any data type, and it is very convenient to read and write the data in the array.

```
// Define an Array
const myArray = [1, "test", true, 10];

// Read data in Array
console.log(myArray[0]); // result is 1
console.log(myArray[1]); // result is test
```

Object

Objects are the way JavaScript describes the world.In real life, when we describe a person through language, we usually start with height, age, etc.: his name is Bob, he is 22 years old this year, and his height is 170. Using JavaScript as a programming language can be described in a more structured way, so let's extract the information expressed above and sort it out:

Attributes	Content
Name	Bob
Age	22
Height	170

An object is actually a special expression of the above structured information by JavaScript. Name, age, etc., are usually called keys (which can be understood as a special variable name), while the specific content is the value:

```
// Define an Object
const people = {
  name: "Bob",
  age: 22,
  height: 170,
};

// Read data in Object
console.log(people.name); // result is Bob
console.log(people.age); // result is 22
```

At this point, we can think of `people` as an object with `name`, `age` and other key information.

Variables and Constants

In JavaScript, both Variables and Constants are used to store data, and the former is similar to variables in algebra-the data it stores can be changed, while the latter is not.

```
let myFalse = false; // Define a Variable: myFalse
const myNumber = 1; // Define a Constant: myNumber, value: 1
const myString = "test"; // Define a Constant: myString, value: 'test'
myFalse = true; // update myFalse, the updated value is true
console.log(myFalse);
console.log(myNumber);
console.log(myString);
```

You can see that the above constants `myNumber` and `myString` start with `const`, followed by a specific name and `=`, and finally write the specific value.Variables are defined in a similar way, but they need to start with `let`, for example, the variable `myFalse` above.Variables defined using `let` can be updated, just like the last `myFalse = true` in the above code to change `myFalse` from false to true.

If you want to learn more about `const` and `let`, I suggest searching in the [recommended tutorial](#Tutorial recommendation) above.

Operator

Computing is one of the most common uses of programming, so how to use operators is also a very important part of learning a language.

Assignment

`=`: This one has been mentioned in the chapter Variables and Constants above

Addition, Subtraction, Multiplication and Division

+

```
// Two numbers of type Number will be added
const myNumber1 = 1;
const myNumber2 = 2;
console.log(myNumber1 + myNumber2); // result is 3

// Two String types of data will be concatenated
const firstName = "Bob";
const lastName = "APITable";
console.log(firstName + " " + lastName); // result is Bob APITable
```

- * /

```
// Usually only used for Number types
const myNumber1 = 1;
const myNumber2 = 2;

console.log(myNumber1 - myNumber2); // result is -1
console.log(myNumber1 * myNumber2); // the result is 2
console.log(myNumber1 / myNumber2); // the result is 0.5
```

Compare

Here are some commonly used operators for reference

`>=`: greater than or equal to

`>`: greater than

`<=`: less than or equal to

`<`: less than

`==`: equal to

`!=`: not equal to

Conditional statement

The if statement is a very common way of conditional judgment in JavaScript, and is usually used in conjunction with comparison operators:

```
const muNumber1 = 1;
const muNumber2 = 2;

if (muNumber1 == muNumber2) {
  console.log(1);
} else if (muNumber1 > muNumber2) {
  console.log(2);
} else {
  console.log(3);
}
// result is 3
```

In addition, JavaScript also supports switch statement for conditional judgment. If you are interested, you can go to Recommended tutorial search in.

Loop statement

If you want to be able to execute the same code repeatedly when writing code, you can use the for statement to achieve it.

```
const myNumber = 0;
const circleNum = 100; // number of cycles

for (i = 0; i < circleNum; i++) {
  myNumber = myNumber + i;
}

console.log(myNumber); // the result is 4950
```

- i=0 is to initialize the loop variable
- i<circleNum is to judge whether the loop variable is less than the defined number of loops, if so, it will continue to execute
- i++ means i will be incremented by 1 after each loop

In addition, JavaScript can also use for/in and While statements to loop. If you are interested, you can go to the Recommended tutorial search in.

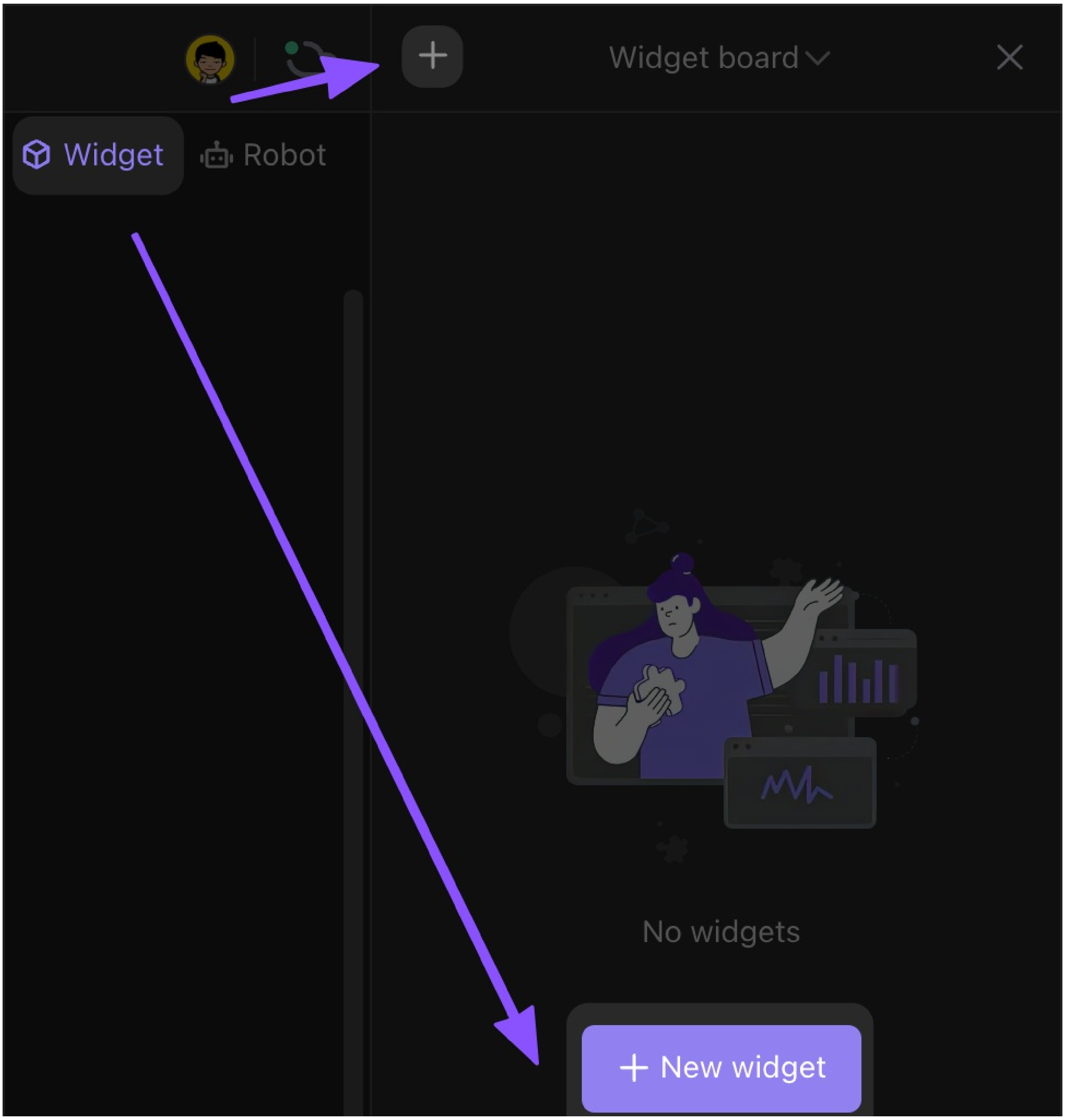
Previous
« Generate latitude and longitude from IP address

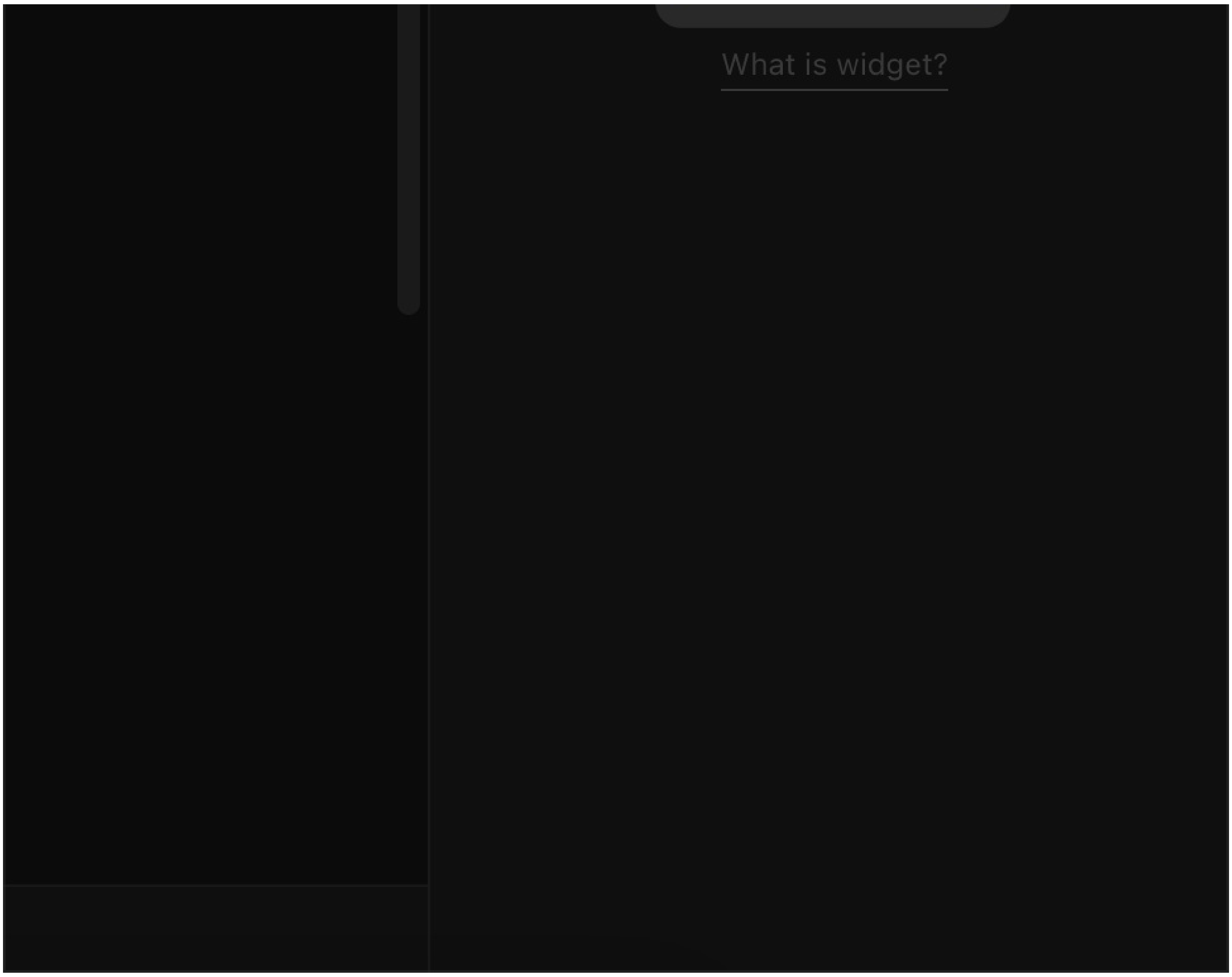
Next
Script API Reference »

How to install

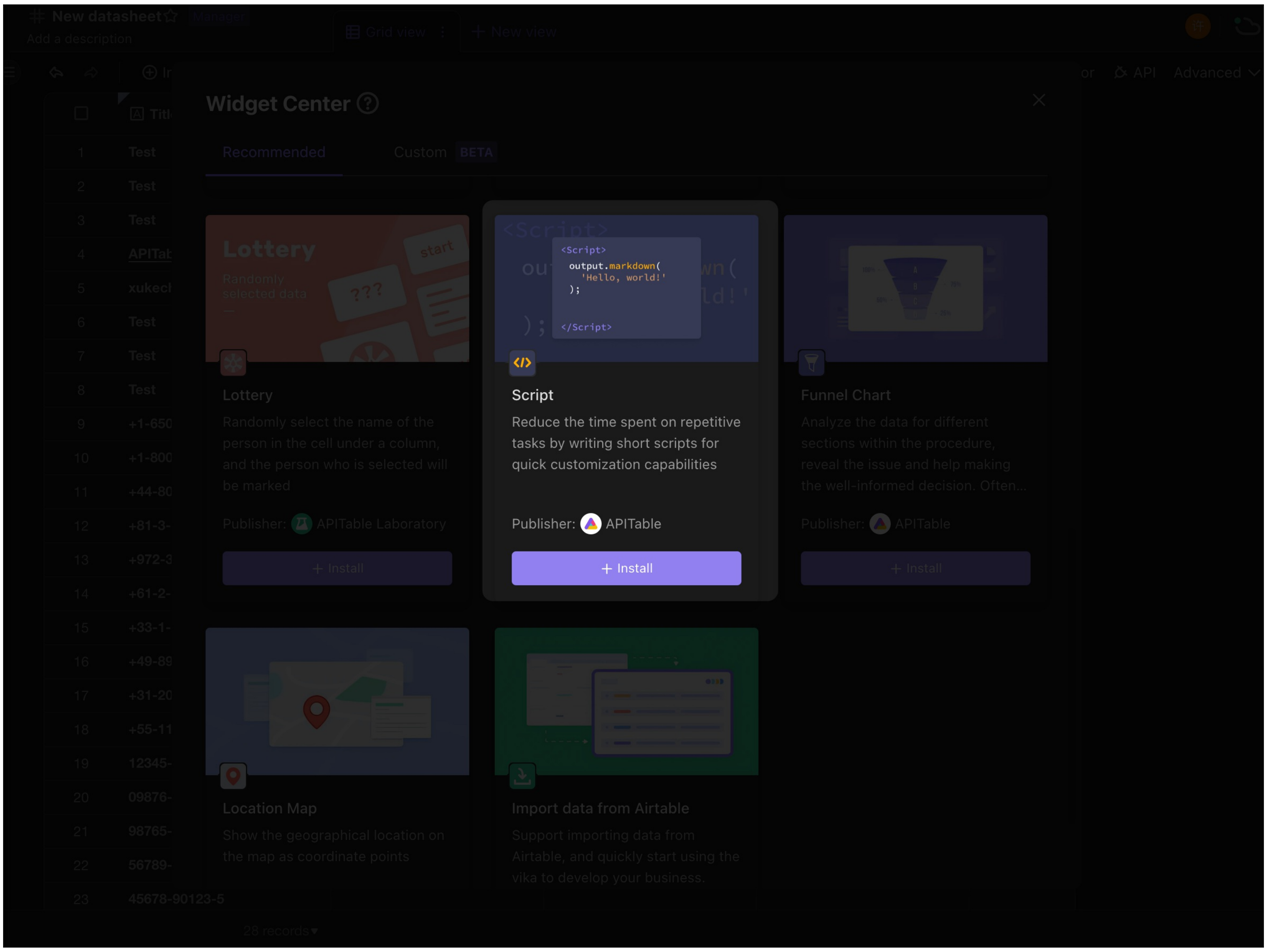
Install the Script widget

To install Script widget, open a datasheet, click the "widget" button on the right to expand the widget panel, and then click the "New widget" button.





Find the "Script" widget in the Widget Center and install it.



Lottery

Randomly select the name of the person in the cell under a column, and the person who is selected will be marked

Publisher: APITable Laboratory

```
<Script>
  output.markdown(
    'Hello, world!'
  );
</Script>
```

Script

Reduce the time spent on repetitive tasks by writing short scripts for quick customization capabilities

Publisher: APITable

Funnel Chart

Analyze the data for different sections within the procedure, reveal the issue and help making the well-informed decision. Often...

Publisher: APITable

Location Map

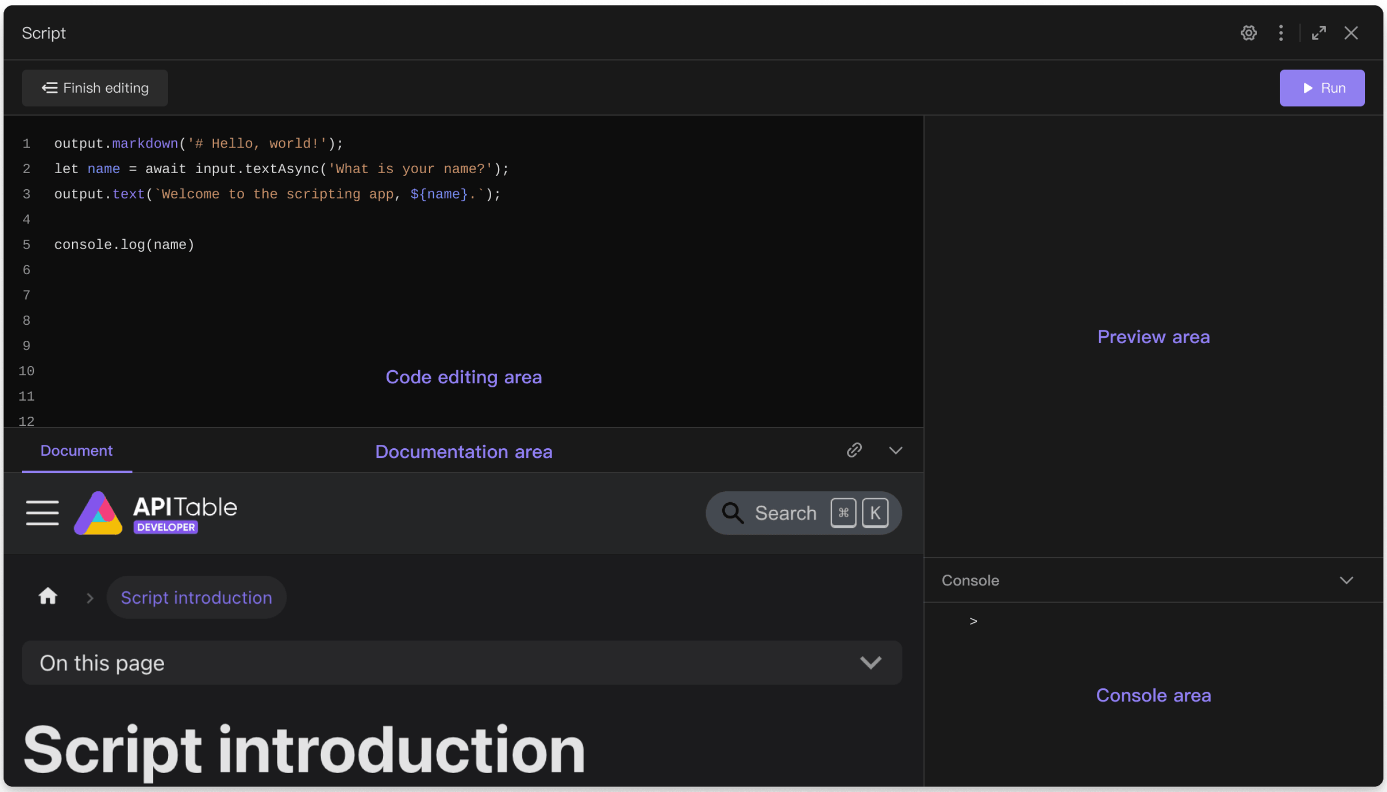
Show the geographical location on the map as coordinate points

Import data from Airtable

Support importing data from Airtable, and quickly start using the vika to develop your business.

How to use the Scripts widget

The Script widget have four functional areas:



```
output.markdown('# Hello, world!');
let name = await input.textAsync('What is your name?');
output.text(`Welcome to the scripting app, ${name}.`);

console.log(name)
```

```
>
```

Script introduction

- Code editing area: the main area for writing and modifying code, supporting smart code prompts
- Documentation area: displaying documents related to Script API, which can be viewed without switching pages
- Preview area: displaying declarative and standardized UI for scripts
- Console area: print the information when the script is running, such as the content of `Console.log()`

Previous
« Getting Started

Next
Development »

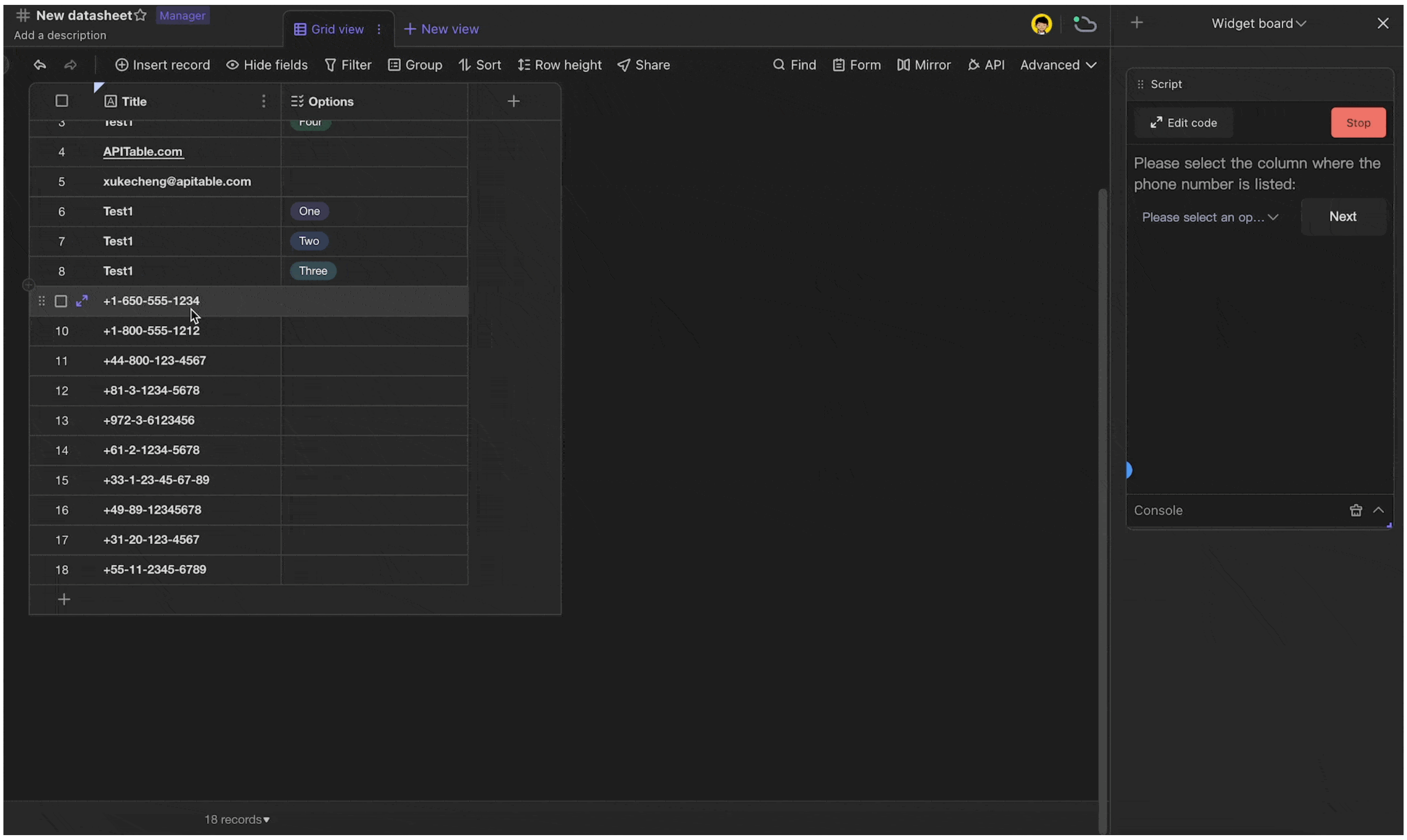
Find the phone number in a column

INFO

Support finding and displaying phone numbers in a specified format in a column of data, such as +1-650-555-1234

DEMO

The following is example data, not real phone number



	Title	Options
3	Test1	Four
4	APITable.com	
5	xukecheng@apitable.com	
6	Test1	One
7	Test1	Two
8	Test1	Three
	+1-650-555-1234	
10	+1-800-555-1212	
11	+44-800-123-4567	
12	+81-3-1234-5678	
13	+972-3-6123456	
14	+61-2-1234-5678	
15	+33-1-23-45-67-89	
16	+49-89-12345678	
17	+31-20-123-4567	
18	+55-11-2345-6789	

Please select the column where the phone number is listed:

Source Code

```
const datasheet = await space.getActiveDatasheetAsync();
const phoneField = await input.fieldAsync("Please select the text column where the phone number is listed:", datash
const phoneFieldId = phoneField.id;

/* This regrex matches phone numbers that start with a + sign,
* followed by three groups of digits separated by either a hyphen - or whitespace.
* The groups can have any number of digits, and the final group must end with at least one digit.
* If you need to match phone numbers in a different format, you can modify the regular expression accordingly. */
const regexReg = /^\+(?:[0-9]+[\-\s]*){3}[0-9]+$/;

const records = await datasheet.getRecordsAsync();
const finalData = ['Matched Phone Number'];

for (let record of records) {
  let cellValue = record.getCellValue(phoneFieldId);

  if (cellValue == null) continue;

  const validation = cellValue.match(regexReg);

  if (validation != null) {
    finalData.push(record.getCellValueString(phoneFieldId));
  }
}

if (finalData.length){
  output.table(finalData);
} else {
  output.text("No matching data")
}
```

Previous
« Find the id card number in a column

Next
Link records with duplicate values »

DEVELOPER

S

Getting StartedExamplesExtract URL from attachment

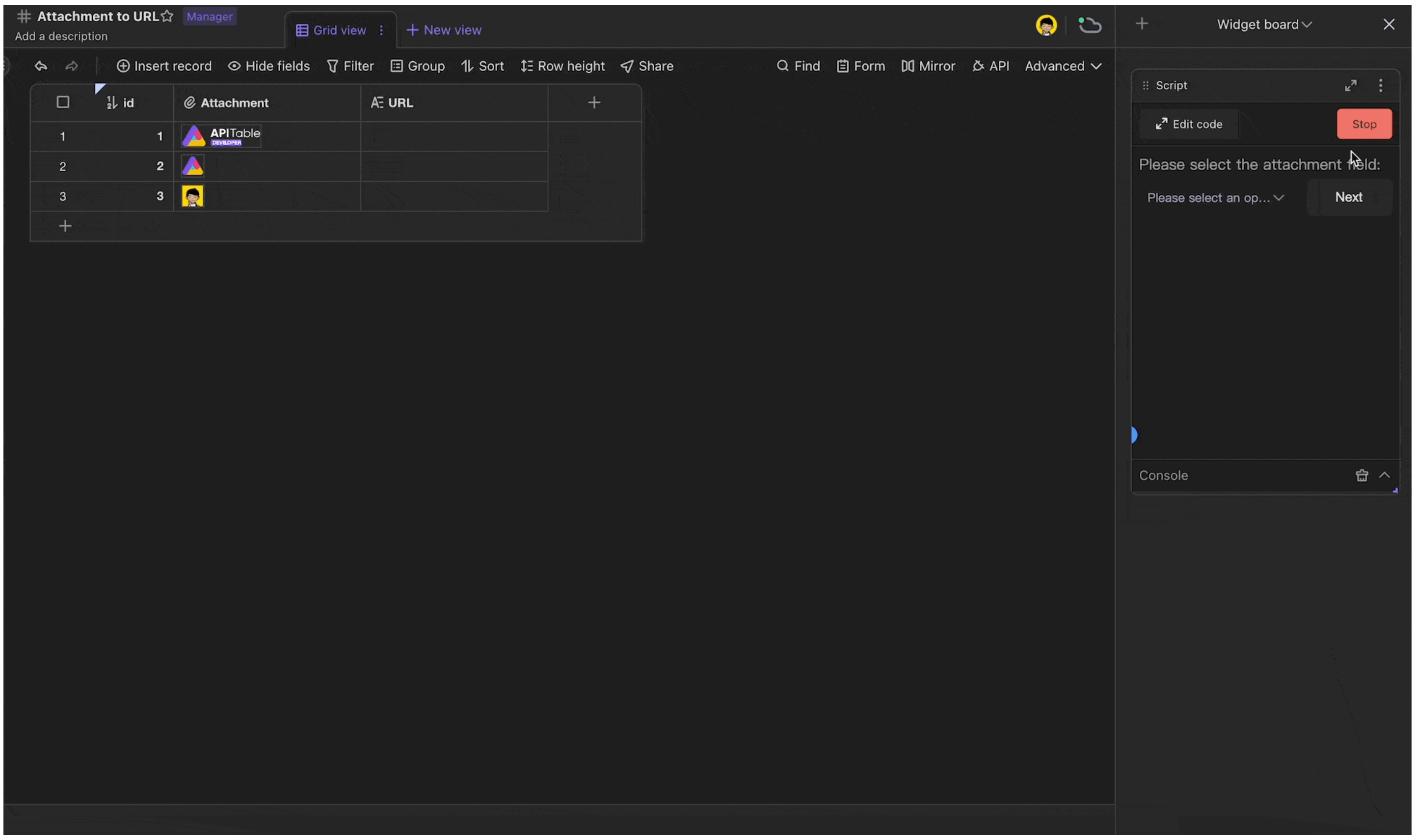
On this page

Extract URL from attachment

INFO

Support converting attachments into links that can be accessed directly

Demo



Source code

```
const datasheet = await space.getActiveDatasheetAsync();

const attachmentField = await input.fieldAsync(
  "Please select the attachment field:",
  datasheet
);
const urlField = await input.fieldAsync(
  "Please select the URL field:",
  datasheet
);

const isCover = await input.textAsync(
  "Whether to overwrite existing URLs (yes/no):"
);

// If the value entered in "Whether to overwrite the existing URL" is not "Yes" or "No", a relevant prompt will be
if (isCover != "yes" && isCover != "no") {
  output.text("Please enter the correct value!!!");
} else {
  const urlFieldId = urlField.id;
  const attachmentFieldId = attachmentField.id;

  const records = await datasheet.getRecordsAsync();

  for (let record of records) {
    const recordId = record.id;
    const attachmentCellValue = record.getCellValue(attachmentFieldId);
    const urlCellValue = record.getCellValueString(urlFieldId);
    //The empty data does not belong to the content to be converted into URL, so it is skipped
    if (attachmentCellValue == null) continue;
    if (isCover == "no" && urlCellValue != null) continue;
    // Get the url value of the attachment
    const url = attachmentCellValue[0].url;
    datasheet.updateRecordAsync(recordId, { [urlFieldId]: url });
  }

  output.text("Attachment converted to URL!!!");
}
```

Previous
« Link records with duplicate values

Next
Generate latitude and longitude from IP address »

DEVELOPER

S

Getting StartedExamplesGenerate latitude and longitude from IP address

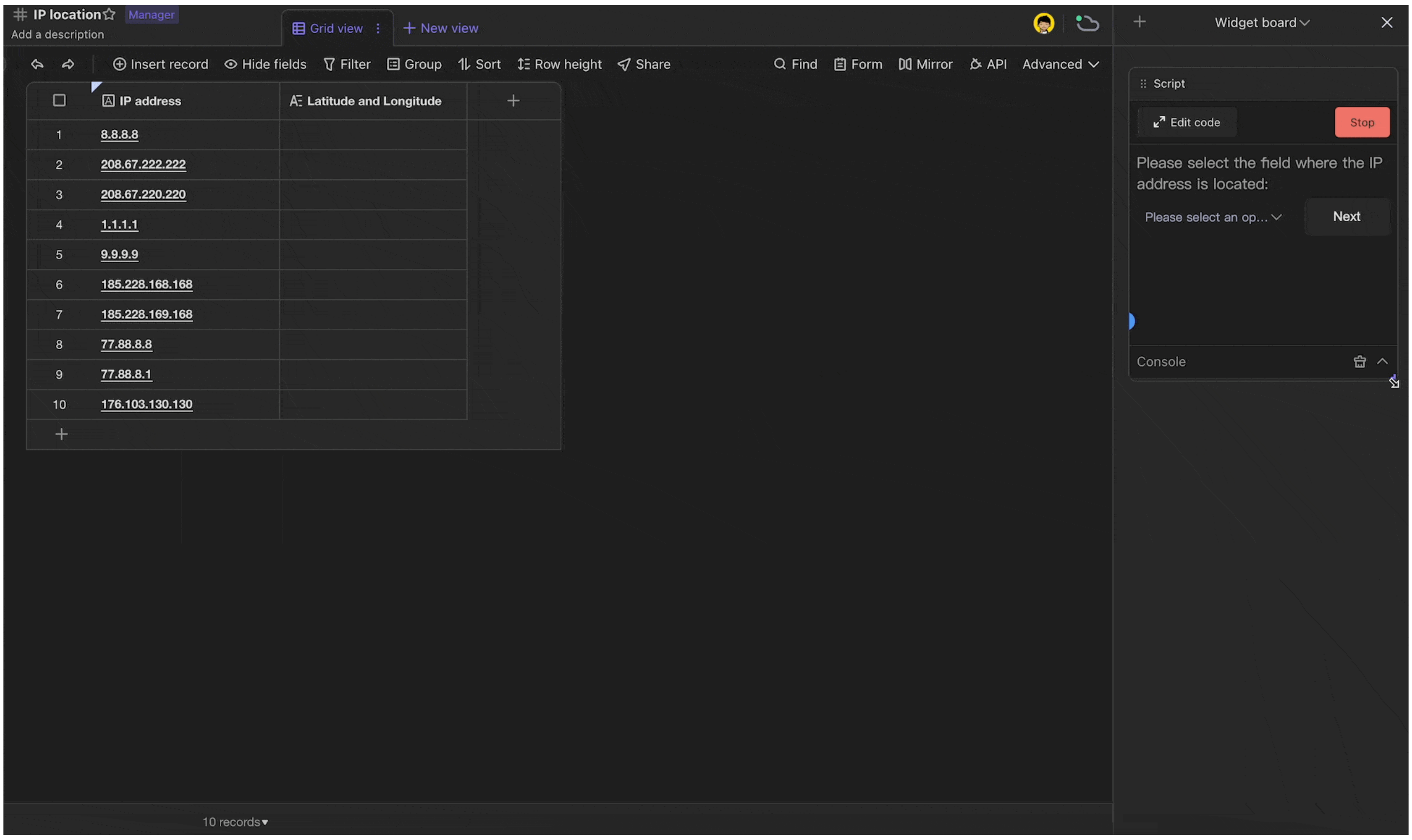
On this page

Generate latitude and longitude from IP address

INFO

Generate the latitude and longitude information of the area based on the IP address

Demo



IP location
Add a description

	IP address	Latitude and Longitude
1	8.8.8.8	
2	208.67.222.222	
3	208.67.220.220	
4	1.1.1.1	
5	9.9.9.9	
6	185.228.168.168	
7	185.228.169.168	
8	77.88.8.8	
9	77.88.8.1	
10	176.103.130.130	

10 records

Please select the field where the IP address is located:

Source code

```
const datasheet = await space.getActiveDatasheetAsync();
const ipField = await input.fieldAsync(
  "Please select the field where the IP address is located:",
  datasheet
);
const coordinateField = await input.fieldAsync(
  "Please select latitude and longitude to generate field:",
  datasheet
);

// Define a dataset to store the data used to update the "latitude and longitude" column
const finalData = [];

// Obtain coordinate information by IP address
async function getCoordinateInfo(ip) {
  const res = await fetch(`https://ipwho.is/${ip}`);
  return await res.json();
}

const ipFieldId = ipField.id;
const coordinateFieldId = coordinateField.id;
const records = datasheet.getRecordsAsync();

for (let record of records) {
  const recordId = record.id;
  let ip = record.getCellValue(ipFieldId);
  //The data that is empty does not belong to the content to be converted into latitude and longitude, so it is ski
  if (ip == null) continue;
  const res = await getCoordinateInfo(ip);
  const coordinate = `${res.longitude}, ${res.latitude}`;
  // Add the data that needs to be updated to the finalData array
  finalData.push({
    id: recordId,
    valuesMap: {
      [coordinateFieldId]: coordinate,
    },
  });
}

if (finalData.length) {
  await datasheet.updateRecordsAsync(finalData);
}

output.text("IP address has been converted to latitude and longitude!!!");
```

[Previous
« Extract URL from attachment]

[Next
Appendix: JavaScript Getting Started Guide »]

DEVELOPER

S

Getting StartedExamplesFind and replace

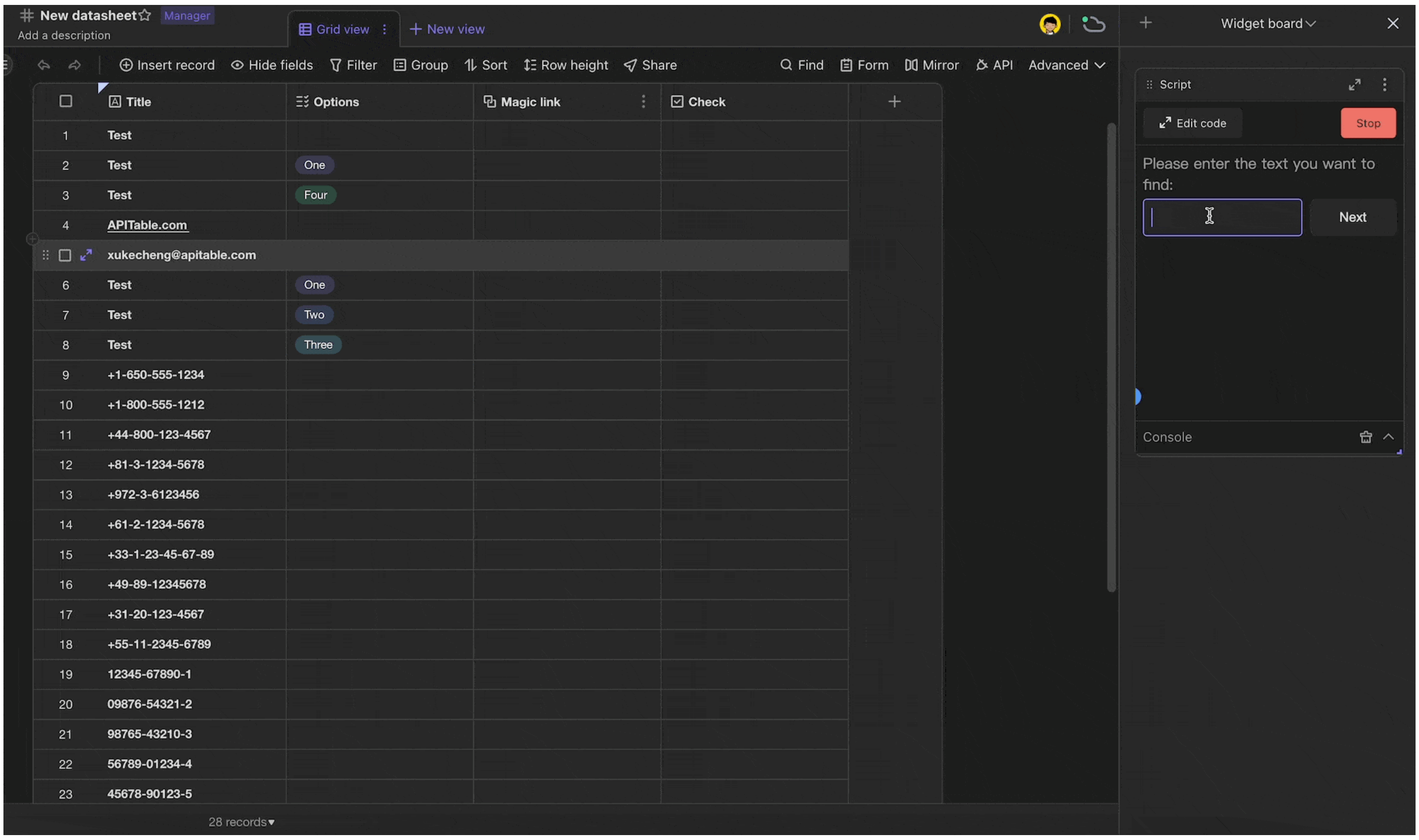
On this page

Find and replace

INFO

Supports finding and replacing specified text in specified fields

Demo


New datasheet Manager
Add a description
Grid view
New view
Widget board

Insert record Hide fields Filter Group Sort Row height Share Find Form Mirror API Advanced

	Title	Options	Magic link	Check
1	Test			
2	Test	One		
3	Test	Four		
4	APITable.com			
	xukecheng@apitable.com			
6	Test	One		
7	Test	Two		
8	Test	Three		
9	+1-650-555-1234			
10	+1-800-555-1212			
11	+44-800-123-4567			
12	+81-3-1234-5678			
13	+972-3-6123456			
14	+61-2-1234-5678			
15	+33-1-23-45-67-89			
16	+49-89-12345678			
17	+31-20-123-4567			
18	+55-11-2345-6789			
19	12345-67890-1			
20	09876-54321-2			
21	98765-43210-3			
22	56789-01234-4			
23	45678-90123-5			

28 records

Script
Edit code
Stop

Please enter the text you want to find:

Next

Console

Source code

```
const findText = await input.textAsync(
  "Please enter the text you want to find:"
);
const replaceText = await input.textAsync(
  "Want to replace the find text with:"
);

// The currently active form needs to be obtained through the Space API and passed to input.fieldAsync as a paramet
const datasheet = await space.getActiveDatasheetAsync();
const field = await input.fieldAsync(
  "Please select the field name you want to find",
  datasheet
);
// Get the id of field
const fieldId = field.id;

const records = await datasheet.getRecordsAsync();
const finalData = [];

// traverse records
for (let record of records) {
  const recordId = record.id;
  // Get the data of the specified field of the record as cellValue
  const cellValue = record.getCellValueString(fieldId);

  // If the acquired data is empty, jump out of this cycle and directly execute the next time
  if (cellValue == null) continue;

  // Replace findText in cellValue with replaceText, and use a new variable - newCellValue
  const newCellValue = cellValue.replaceAll(findText, replaceText);
  // Determine whether the data before replacement is consistent with the data after replacement
  if (cellValue !== newCellValue) {
    // Inconsistency means that the corresponding record in the table needs to replace cellValue with newCellValue,
    finalData.push({
      id: recordId,
      valuesMap: { [fieldId]: newCellValue },
    });
  }
}

// Determine whether there is data in finalData, if there is no data, there is no need to replace
if (finalData.length) {
  await datasheet.updateRecordsAsync(finalData);
  output.text("The replacement is complete!");
} else {
  output.text("No data found to be replaced");
}
```

Previous
« Code example

Next
Data matcher »

Getting Started

[Previous
« Script introduction]

[Next
How to install »]

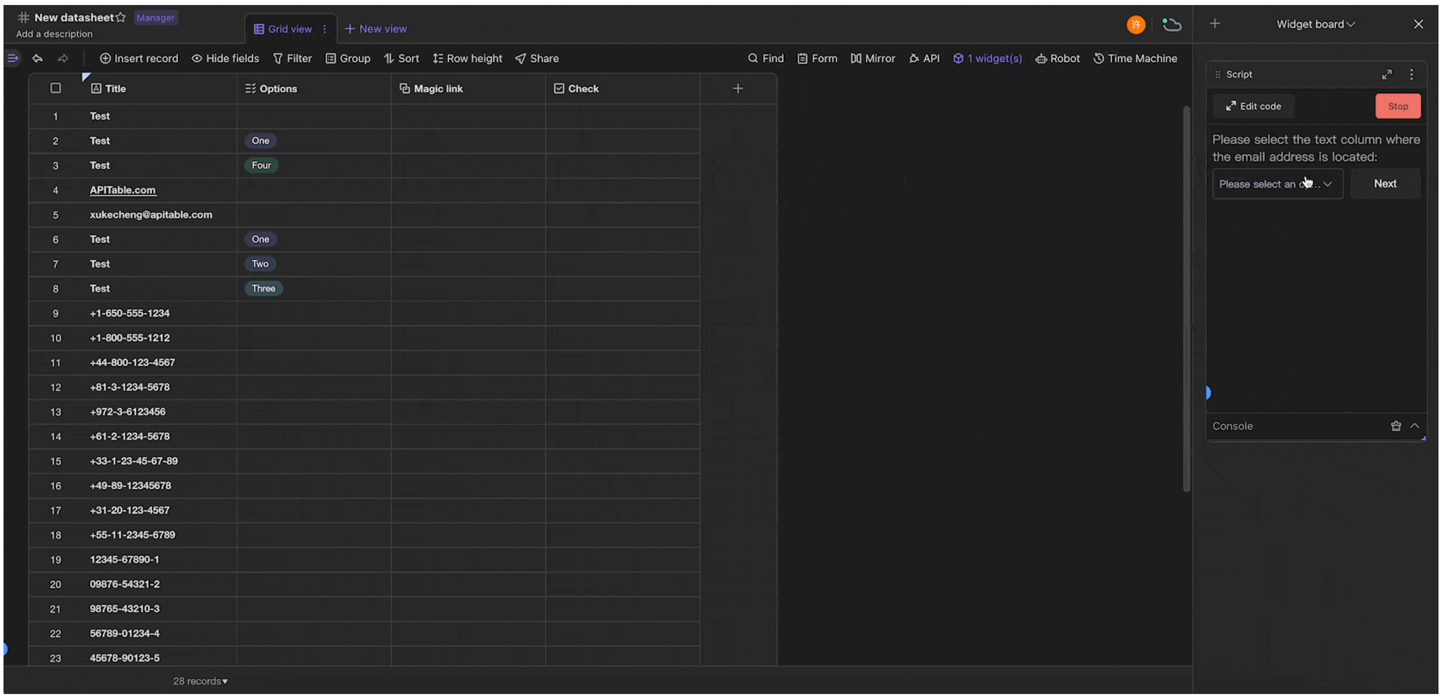
Verify whether the email is legal

INFO

Supports checking whether the email meets the specified requirements in the text type column, and if it does, it will be marked as √ in the corresponding checkbox column

Demo

The following is example data, not real email


New datasheet Manager
Add a description
Grid view | New view
Insert record | Hide fields | Filter | Group | Sort | Row height | Share | Find | Form | Mirror | API | 1 widget(s) | Robot | Time Machine

	Title	Options	Magic link	Check
1	Test			
2	Test	One		
3	Test	Four		
4	APITable.com			
5	xukecheng@apitable.com			
6	Test	One		
7	Test	Two		
8	Test	Three		
9	+1-650-555-1234			
10	+1-800-555-1212			
11	+44-800-123-4567			
12	+81-3-1234-5678			
13	+972-3-6123456			
14	+61-2-1234-5678			
15	+33-1-23-45-67-89			
16	+49-89-12345678			
17	+31-20-123-4567			
18	+55-11-2345-6789			
19	12345-67890-1			
20	09876-54321-2			
21	98765-43210-3			
22	56789-01234-4			
23	45678-90123-5			

28 records

Please select the text column where the email address is located:

Source code

```
const datasheet = await space.getActiveDatasheetAsync();
const mailField = await input.fieldAsync("Please select the text column where the email address is located:", datas
const mailFieldId = mailField.id;

const checkField = await input.fieldAsync("Please select the check field, which needs to be a check type:", datashe
const checkFieldId = checkField.id;

const records = await datasheet.getRecordsAsync();

const mailRule = await input.textAsync('Please enter the email suffix, such as @aitable.ai:');
const mailReg = new RegExp(mailRule);
const finalData = [];

for (let record of records) {
  const recordId = record.id;
  let cellValue = record.getCellValue(mailFieldId);

  if (cellValue == null) continue;

  const validation = cellValue.match(mailReg);
  if (validation != null) {
    finalData.push({
      id: recordId,
      valuesMap: { [checkFieldId]: true }
    });
  }
}

if (finalData.length) {
  await datasheet.updateRecordsAsync(finalData);
}

output.text('Complete the check!!!')
```

Previous
« Data matcher

Next
Find the id card number in a column »

Script introduction

Script is an extension of the AITable, which supports you program online and interact with the datasheets through the built-in Script API. With this ability, you can quickly complete all kinds of repetitive work and greatly improve your work efficiency. For example, you can:

- Program to generate your own AITable widget that supports **find and replace** in less than 30 lines of code.
- Program a script to verify the data in the datasheet according to custom rules, such as verifying whether the email address is valid, and finding specified ID number or mobile phone number.
- Program a script to call any API quickly to support GET, POST and other request methods.
- ...

Script helps you **complete code writing** faster to complete all kinds of repetitive and complicated tasks in daily work and life, achieving **fast delivery**!

At the same time, it is also very friendly to developer starters. You just need to be familiar with **JavaScript** and have a **browser** to start!

Difference with widget

If experienced developers see the scope listed above, they may have some questions: It seems that the widget SDK can be done. Is there any difference between the two ways?

Scenes	Script	Widget
Enter a lookup value and replace it with the value I want.	**Experienced developers can complete it in less than 1 hour**, with a standardized UI interface.	Experienced developers will take about **1 day** from 0 to build their own development environment and think about UI layout and interaction.
Convert attachments to URLs or URL downloads to attachments	**Experienced developers can complete it in less than 2 hours**, with a standardized UI interface.	Experienced developers need about **0.5 days** from 0 to build their own development environment and think about UI layout and interaction.
Make a dynamic and interactive chart.	It cannot be achieved, only a declarative standardized UI interface.	It can be achieved, can be fully customized, and can be made very beautifully.
Embed third-party services into the AITable	It cannot be achieved, only a declarative standardized UI interface.	It can be achieved, can be fully customized, and can be made very beautifully.

In terms of scenarios, scripts can complete logical computing tasks and standardized UI rendering faster, while widgets are more suitable for complex and customized UIs and interactions.

What do I need to prepare before developing the script?

- Familiar with the JavaScript
- Prepare a Chromium browser, such as Google Chrome, Microsoft Edge

I don't know programming at all. How to get started?

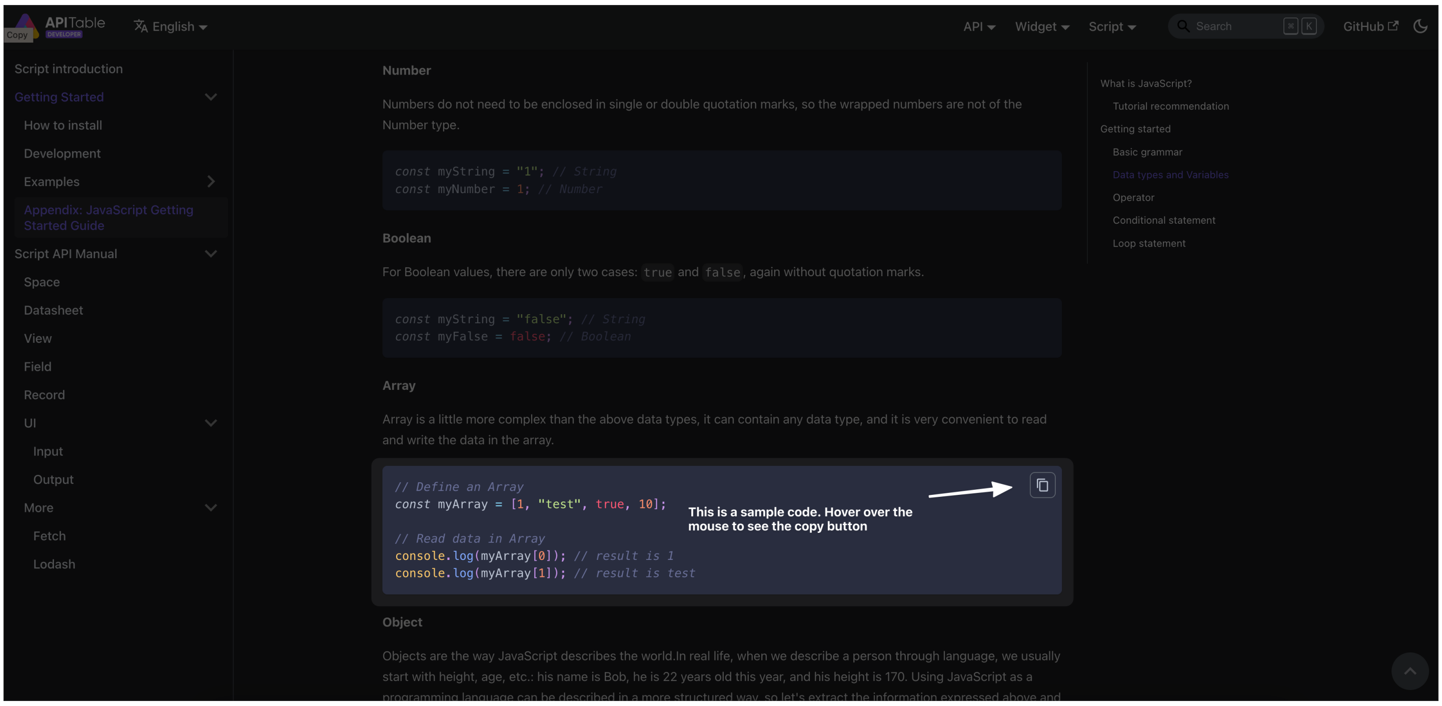
As mentioned above, script is easier to get started than widget. It is recommended that beginners or programming beginners can learn according to the following paths:

1. Installation to learn how to install the script widget and its corresponding functional partition.
2. Practice will teach you how to use JavaScript combined with the Script API to make a search and replacement widget with less than 30 lines of code.
3. If you are not familiar with JavaScript, it is recommended to familiarize yourself with the basic JavaScript syntax through the Appendix: JavaScript Getting Started Guide

How to run sample code

If you are going to learn how to use scripts according to the above documents, you will find that each chapter contains a large amount of sample code, which can help you learn faster.

First move the mouse into the sample code area to copy it, as shown in the following figure:



Number

Numbers do not need to be enclosed in single or double quotation marks, so the wrapped numbers are not of the Number type.

```
const myString = "1"; // String
const myNumber = 1; // Number
```

Boolean

For Boolean values, there are only two cases: `true` and `false`, again without quotation marks.

```
const myString = "false"; // String
const myFalse = false; // Boolean
```

Array

Array is a little more complex than the above data types, it can contain any data type, and it is very convenient to read and write the data in the array.

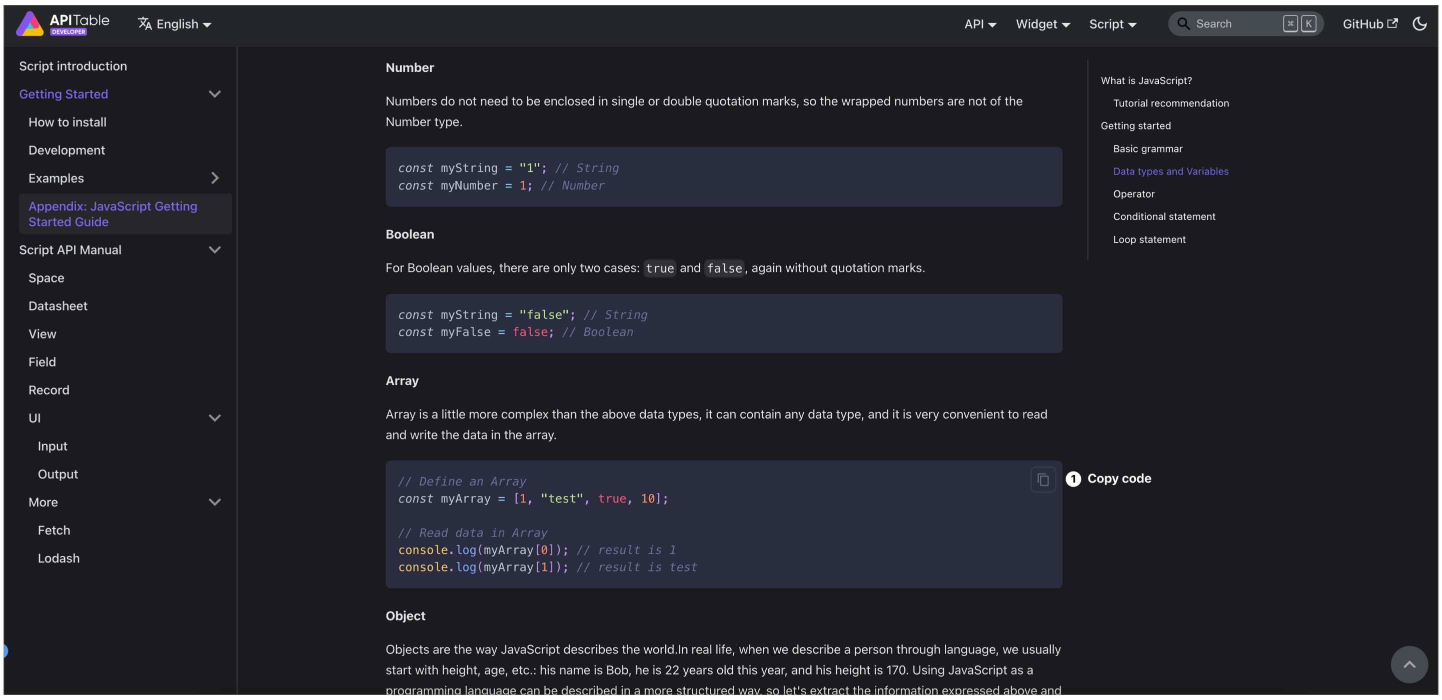
```
// Define an Array
const myArray = [1, "test", true, 10];

// Read data in Array
console.log(myArray[0]); // result is 1
console.log(myArray[1]); // result is test
```

This is a sample code. Hover over the mouse to see the copy button

Object

Objects are the way JavaScript describes the world.In real life, when we describe a person through language, we usually start with height, age, etc.: his name is Bob, he is 22 years old this year, and his height is 170. Using JavaScript as a programming language can be described in a more structured way, so let's extract the information expressed above and

The way you run the sample code can refer to the following process:



Number

Numbers do not need to be enclosed in single or double quotation marks, so the wrapped numbers are not of the Number type.

```
const myString = "1"; // String
const myNumber = 1; // Number
```

Boolean

For Boolean values, there are only two cases: `true` and `false`, again without quotation marks.

```
const myString = "false"; // String
const myFalse = false; // Boolean
```

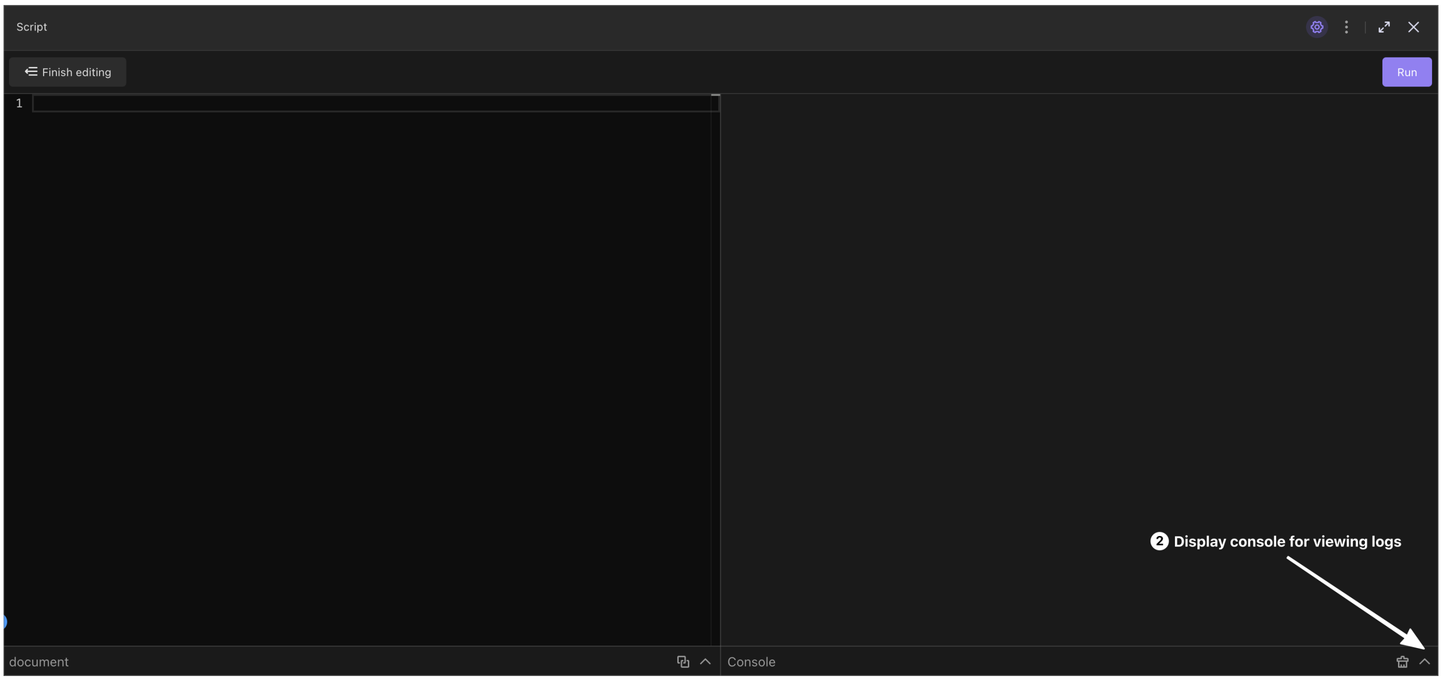
Array

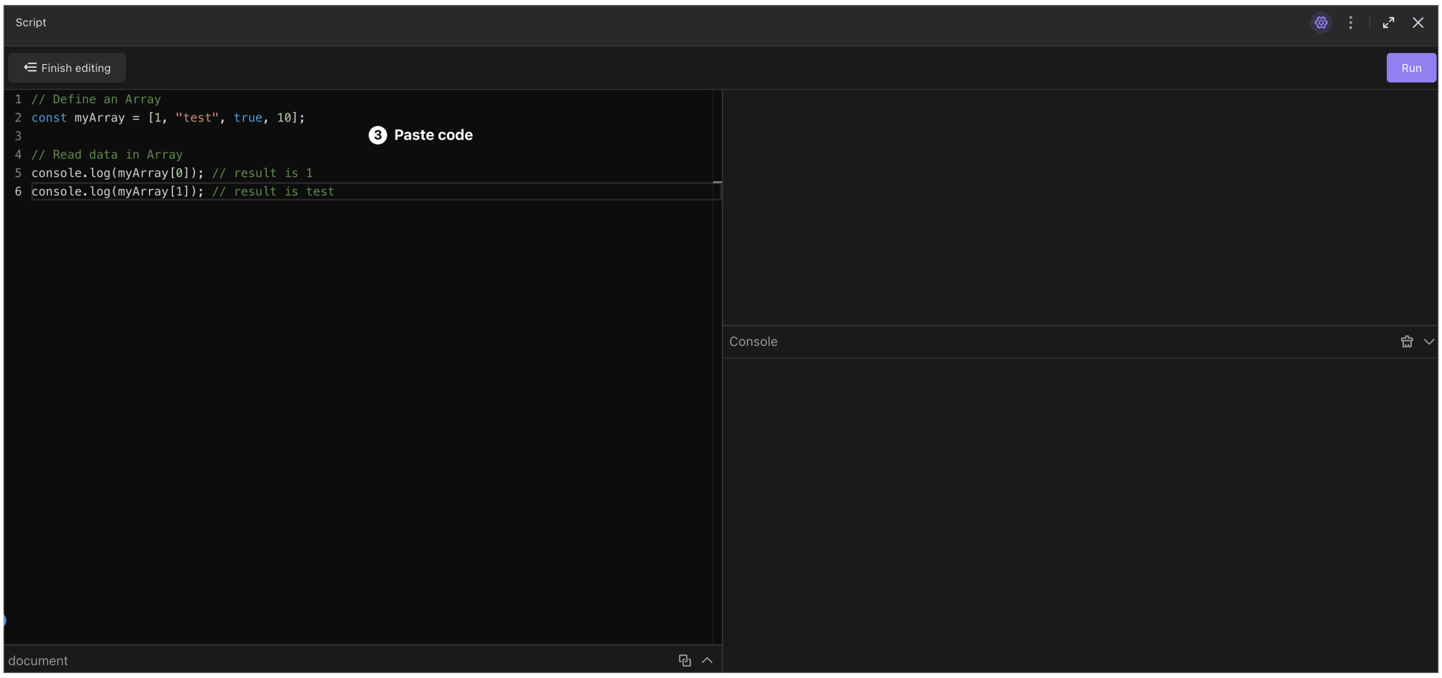
Array is a little more complex than the above data types, it can contain any data type, and it is very convenient to read and write the data in the array.

```
// Define an Array
const myArray = [1, "test", true, 10];

// Read data in Array
console.log(myArray[0]); // result is 1
console.log(myArray[1]); // result is test
```

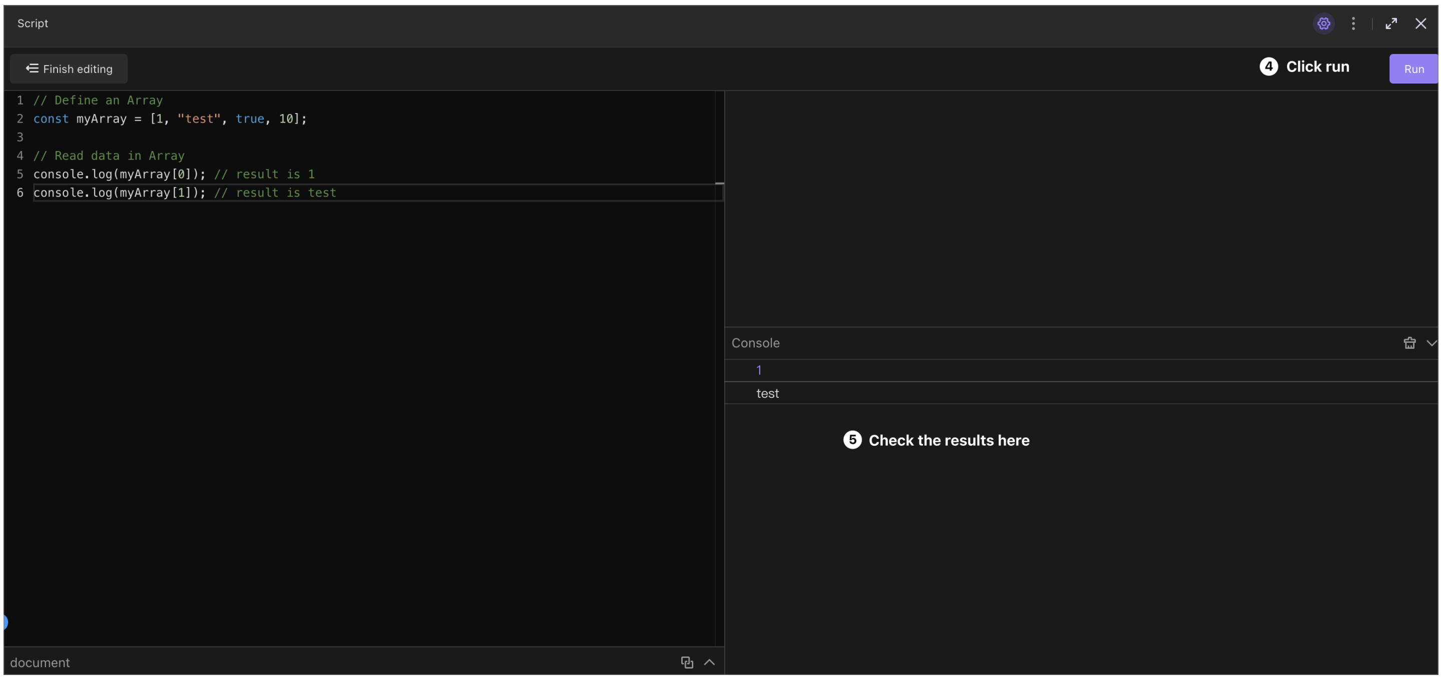
Object

Objects are the way JavaScript describes the world.In real life, when we describe a person through language, we usually start with height, age, etc.: his name is Bob, he is 22 years old this year, and his height is 170. Using JavaScript as a programming language can be described in a more structured way, so let's extract the information expressed above and





```
// Define an Array
const myArray = [1, "test", true, 10];

// Read data in Array
console.log(myArray[0]); // result is 1
console.log(myArray[1]); // result is test
```



```
// Define an Array
const myArray = [1, "test", true, 10];

// Read data in Array
console.log(myArray[0]); // result is 1
console.log(myArray[1]); // result is test
```

```
1
test
```

Next
Getting Started »