
Mesh Spreadsheet

Mesh is a JavaScript code editor that feels like a spreadsheet.

Specifically, Mesh is a spreadsheet UI wrapper around a text file editor. Actions on the grid are automatically translated to changes in the JavaScript code.

Mesh helps people maximise their personal productivity and to share their work with others. Consider Mesh if you:

- use JavaScript, but want rapid visual feedback and a convenient grid UI
- use spreadsheets, but feel constrained by Excel's limitations.

WARNING! Mesh is under active development. The UI and APIs will likely change a lot, it is not well optimised, and there are lots of missing features and bugs. Sign up for the mailing list and follow the project on Twitter at @MeshSpreadsheet.

Video demos (alpha)

- Basics
- Deep-dive on tables

Loan calculator example:

Name-based referencing

When you create a cell, it gets given a name based on its location (eg `D4`). You can press `F3` while selecting the cell to toggle whether its name is shown, and can edit the name by typing over that name like any other value.

What about nested data? The contents of objects, arrays and tables don't get their own cell references - only the object itself. This ensures cells always have a unique name, and that any visual collisions that arise when using expanding dynamic data don't have an impact on calculations (ie `#SPILL!` errors).

To refer to an element of an array or object, or a table row, use standard property access syntax (eg `=D4[1]` or `=D4.foo` or `=D4[2].someHeading`).

Values and formulas

To overwrite a cell's contents, just select it and start typing. You'll see your edits appear in the formula bar at the top. Press `Enter` when you're done. If you instead want to edit the cell's contents in-place, press `F2` before you start typing.

To make data entry easier, we rewrite your "common sense" input to a JavaScript equivalent if necessary. For example, `Hello world!` (without quotes) will be written into `"Hello world!"` (with quotes). Values can be:

- Numbers: `123`
- Strings: `Hello world!`
- Booleans: `true`
- Functions: `function(){return 1 + 2}`
- Dates: `2018-11-10`
- Regular expressions: `/hello world/g`

Formulas are a way to write "raw" JavaScript code, including code that references other cells. You can add a formula to a cell in the same way as you'd add a value, but with a `=` at the front:

- `=1+2`
- `=some_cell_name + 123.`

Tables

Tables are common in spreadsheets, but less so in traditional programming languages. Mesh hopes to change that!

Tables appear in the grid as rows and columns with a header row at the top.

But in JavaScript terms, tables are arrays of objects. The wrapper function `_makeTable` adds the ability to:

- specify default columns (default row object properties), including calculated columns (parameter 1)
- expand or shrink based on a specified length, including one calculated from other cells at runtime (parameter 2)
- reference other cells in the table based on the row's absolute or relative position, via a few default row properties:
 - `this` (the row)
 - `i` (the row's index)
 - `t` (the table).

Create a table by clicking on an empty cell or value cell and pressing `Ctrl + Alt + t`.

You can then add or delete columns or rows:

- Append to an array or object by typing into the 'new row' cells that appear
- Insert a new cell above a selected cell with `Ctrl =`
- Delete a selected cell with `Ctrl -`.

Delete the table entirely with `Ctrl _` (ie `Ctrl Shift -`).

Search a table for a row with the built-in `find` function, or query it with standard array methods like `map`, `filter` and `reduce`.

Use tables to simplify heavily nested (indented) code, such as complex conditional logic or multi-dimensional concepts. Peter Kankowski's Code Project article [Table-driven approach](#) has some great examples.

Mesh files

Mesh files are just plain-text JavaScript files. To see the contents of the file you're writing, toggle the **code pane** with the link in the status bar at the bottom of the screen.

You can edit the spreadsheet and see the change in the code on the right.

You'll also see some **boilerplate code** near the bottom. The boilerplate:

- turns the cells specification in `_CELLS` into a functioning spreadsheet, with caching of cell values whose inputs have not changed
- provides table creation (with their own internal caching) and the `find` function to query tables alongside standard ES5 JavaScript array methods
- uses a function `sc` to strip out parts of cells that won't make it through the structured clone algorithm when the spreadsheet output is sent through [nb: this may be moved out of the boilerplate]

We're also considering adding: - a ES5-friendly way to write functions in shorter syntax - some standard 'data aggregation' functions like `SUM`.

The boilerplate will be updated from time to time; for this reason, it has a version date in the comments.

Messaging

Mesh sheets are JavaScript apps in their own right, and with a little extra boilerplate, they can send and receive messages.

In particular, you can send in streams of values and get calculations out (demo GIF).

We're looking into the best way to allow Mesh sheets to support a wide variety of message formats. Right now the Mesh app uses cross-document messaging via `onmessage` and `postMessage`.

We're also looking into ways that Mesh sheets can:

- be embedded into other apps
- act as cloud APIs.

Benefits of Mesh (or “a new spreadsheet program? you're crazy!”)

Excel, Google Sheets and LibreOffice are mature, feature-packed products that are unlikely to be displaced anytime soon, particularly where a team is happy using tools they already have and know.

Nevertheless, Mesh has some advantages.

Mesh uses JavaScript as its formula language (a) to bring the power of 'traditional' programming to spreadsheets and (b) because pretty much everyone has access to ES5 JavaScript, even if their computer is old or locked down by IT.

In particular, Mesh offers a first-class user experience in Internet Explorer 11 (which comes bundled with Windows).

Because Mesh is small, free and highly compatible, you can put it on a USB or email it to a friend and have confidence that they can open your sheet files. Alternatively, you can embed your sheet into the Mesh URL as a querystring (*currently broken, but will be fixed soon!*).

Mesh is designed from the ground up to support formulas that output data of arbitrary length. Traditional spreadsheets give all values on the grid a location-based reference, so they can lead to expanding dynamic arrays generating #SPILL! errors that cause calculations to fail.

And finally, the Mesh sheet format is just JavaScript code in a text file, so: - `diffing` is easy (function is built into Windows: `FC` in CMD, `Compare-Object` in PowerShell - you could even use Word's compare tool in a pinch) - it integrates with standard version control systems like Git - you don't need Mesh to run a Mesh file, so you can integrate Mesh files into other systems, including webpages (see the messaging discussion above).

Known issues

- Incomplete syntax support (eg spreads ([...`elements`, "`extra`"]); Maps and Sets; some functions)
- When editing in the formula bar, it should show the edits in the cell 'live', highlight input cells in the grid, and potentially show dependence/precedence arrows
- Poor compatibility with standard data formats like CSV (I need to figure out how to integrate a CSV parser without compromising the ability to run Mesh files without Mesh)
- Poor integration with Electron (such as menu items for New, Open, Save, Save As)
- There's no clear idiomatic way to import external libraries
- Poor JavaScript standard library (although we have some ideas to solve this).

I want to contribute!

Awesome! Please check out the contributions guide.

The codebase is pretty rough right now, so feel free to get in touch via Twitter if you have any questions.

Bugs, issues, enhancements, contact

Please file any bugs, issues or enhancements via GitHub.

Contact me at Twitter: @MeshSpreadsheet.