Getting Started

This getting started describes how to install, load, and use math.js.

Install

Math.js can be installed using various package managers like npm, or by just downloading the library from the website: https://mathjs.org/download.html.

To install via npm, run:

```
npm install mathjs
```

Other ways to install math.js are described on the website.

Load

Math.js can be used in node.js and in the browser. The library must be loaded and instantiated. When creating an instance, one can optionally provide configuration options as described in Configuration.

ES modules

Load the functions you need and use them:

```
import { sqrt } from 'mathjs'

console.log(sqrt(-4).toString()) // 2i
```

To use lightweight, number only implementations of all functions:

```
import { sqrt } from 'mathjs/number'

console.log(sqrt(4).toString()) // 2
console.log(sqrt(-4).toString()) // NaN
```

You can create a mathjs instance allowing configuration and importing of external functions as follows:

```
import { create, all } from 'mathjs'

const config = { }
const math = create(all, config)

console.log(math.sqrt(-4).toString()) // 2i
```

How to optimize your bundle size using tree-shaking is described on the page Custom bundling.

Node.js

Load math.js in node.js (CommonJS module system):

```
const { sqrt } = require('mathjs')

console.log(sqrt(-4).toString()) // 2i
```

Browser

Math.js can be loaded as a regular JavaScript file in the browser, use the global variable `math` to access the libary once loaded:

```
<!DOCTYPE HTML>
<html>
<head>
  <script src="math.js" type="text/javascript"></script>
</head>
<body>
  <script type="text/javascript">
    console.log(math.sqrt(-4).toString()) // 2i
  </script>
</body>
</html>
```

Use

Math.js can be used similar to JavaScript's built-in Math library. Besides that, math.js can evaluate expressions (see Expressions) and supports chaining (see Chaining).

The example code below shows how to use math.js. More examples can be found in the section Examples.

```
// functions and constants
math.round(math.e, 3)            // 2.718
math.atan2(3, -3) / math.pi      // 0.75
math.log(10000, 10)              // 4
math.sqrt(-4)                    // 2i
math.pow([[-1, 2], [3, 1]], 2)   // [[7, 0], [0, 7]]

// expressions
math.evaluate('12 / (2.3 + 0.7)')    // 4
math.evaluate('12.7 cm to inch')     // 5 inch
math.evaluate('sin(45 deg) ^ 2')     // 0.5
math.evaluate('9 / 3 + 2i')          // 3 + 2i
math.evaluate('det([-1, 2; 3, 1])')  // -7

// chained operations
math.chain(3)
    .add(4)
    .multiply(2)
    .done() // 14
```

Next

To learn more about math.js, check out the available documentation and examples:

- Documentation
- Examples

Examples

- Algebra
- Basic usage
- Bignumbers
- Chaining
- Complex numbers
- Expressions
- Fractions
- Import
- Matrices
- Objects
- Serialization
- Sparse matrices
- Units

Browser examples

- Angle configuration
- Basic usage
- Currency conversion
- Custom separators
- Plot
- Pretty printing with mathjax
- Printing html
- Requirejs loading
- Rocket trajectory optimization
- Webworkers

Advanced examples

- Convert fraction to bignumber
- Custom argument parsing
- Custom datatype
- Custom evaluate using factories
- Custom evaluate using import
- Custom loading
- Custom relational functions
- Custom scope objects
- Expression trees
- Function transform
- More secure eval
- Use bigint
- Web server

Documentation

Math.js is an extensive math library for JavaScript and Node.js. It features a flexible expression parser with support for symbolic computation, comes with a large set of built-in functions and constants, and offers an integrated solution to work with different data types like numbers, big numbers, complex numbers, fractions, units, and matrices.

Math.js can be used in the browser, in node.js and in any JavaScript engine. Installation and download instructions are available on the Download page of the website.

Getting Started

- Getting Started
- Examples

Documentation

- **Core**
  - Configuration
  - Chaining
  - Extension
  - Serialization
- **Expressions**
  - Parsing and evaluation
  - Syntax
  - Expression trees
  - Algebra
  - Customization
  - Security
- **Data Types**
  - Numbers
  - BigNumbers
  - Fractions
  - Complex Numbers
  - Matrices
  - Units
- **Reference**
  - Classes
  - Functions
  - Constants
- Custom bundling
- Command Line Interface
- History

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>math.js | plot</title>
  <script src="https://unpkg.com/mathjs@9.5.0/lib/browser/math.js"></script>

  <script src="https://cdn.plot.ly/plotly-1.35.2.min.js"></script>

  <style>
    input[type=text] {
      width: 300px;
    }
    input {
      padding: 6px;
    }
    body, html, input {
      font-family: sans-serif;
      font-size: 11pt;

    }
    form {
      margin: 20px 0;
    }
  </style>
</head>
<body>

<form id="form">
  <label for="eq">Enter an equation:</label>
  <input type="text" id="eq" value="4 * sin(x) + 5 * cos(x/2)" />
  <input type="submit" value="Draw" />
</form>

<div id="plot"></div>

<p>
  Used plot library: <a href="https://plot.ly/javascript/">Plotly</a>
</p>

<script>
  function draw() {
    try {
      // compile the expression once
      const expression = document.getElementById('eq').value
      const expr = math.compile(expression)

      // evaluate the expression repeatedly for different values of x
      const xValues = math.range(-10, 10, 0.5).toArray()
      const yValues = xValues.map(function (x) {
        return expr.evaluate({x: x})
      })
```

```
      // render the plot using plotly
      const trace1 = {
        x: xValues,
        y: yValues,
        type: 'scatter'
      }
      const data = [trace1]
      Plotly.newPlot('plot', data)
    }
    catch (err) {
      console.error(err)
      alert(err)
    }
  }

  document.getElementById('form').onsubmit = function (event) {
    event.preventDefault()
    draw()
  }

  draw()
</script>

</body>
</html>
```