

# MathJax and Custom Elements



Jan Marthedal Rasmussen

<http://janmr.com>

@janmarthedal

# MathJax

$$f(a) = \frac{1}{2\pi i} \oint_{\gamma} \frac{f(z)}{z-a} dz$$

## Beautiful math in all browsers

A JavaScript display engine for mathematics that works in all browsers.

No more setup for readers. It just works.



QUESTIONS

TAGS

USERS

BADGES

UNANSWERED

ASK QUESTION

## Different methods to compute $\sum_{k=1}^{\infty} \frac{1}{k^2}$

↑  
334  
↓  
★  
230

As I have heard people did not trust Euler when he first discovered the formula (solution of the [Basel problem](#))

$$\zeta(2) = \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}.$$

However, Euler was Euler and he gave other proofs.

I believe many of you know some nice proofs of this, can you please share it with us?

(sequences-and-series) (number-theory) (fourier-analysis) (big-list) (faq)

share cite improve this question

edited Aug 13 at 4:02

community wiki  
14 revs, 8 users 27% AD.

asked 4 years ago  
viewed 34869 times  
active 29 days ago



Love this site?

Get the **weekly newsletter!**

- Top questions and answers
- Important announcements
- Unanswered questions

# Demo: Dynamic math with MathJax

<demo/dynamic-math.html>

# Dynamic math with MathJax

```
var elem = DynMath.createElement('a^2+b^2=c^2');  
container.appendChild(elem);  
DynMath.typeset(elem);
```

# Custom Elements



# Registering a Custom Element

- `document.createElement('my-element', ...)`
- Tag name must
  - be a legal XML tag name
  - contain a '-'
  - not contain upper-case letters

# Registering a Custom Element

- `document.createElement('my-element', ...)`
- Tag name must
  - be a legal XML tag name
  - contain a '-'
  - not contain upper-case letters
  - not be any of `annotation-xml`, `color-profile`, `font-face`, `font-face-src`, `font-face-uri`, `font-face-format`, `font-face-name`, `missing-glyph`



# So what?

```
<head>
  <style>
    my-element { color: red; }
  </style>
</head>
<body>
  <p>Some paragraph with an <my-element>unknown</my-element>
    element</p>

  <my-element></my-element>

  <p>Another paragraph with a <my-element>mysterious</my-element>
    element</p>
</body>
```

# Demo: Unknown element

`demo/unknown.html`

... and why?

# Semantics for humans

```
<google-map latitude="37.774" longitude="-122.419" fit-to-markers>  
  <google-map-marker latitude="37.779" longitude="-122.3892"  
    draggable="true" title="Go Giants!"></google-map-marker>  
  <google-map-marker latitude="37.777" longitude="-122.389">  
    </google-map-marker>  
</google-map>
```

<https://elements.polymer-project.org/elements/google-map>

# Semantics for machines?

# Lifecycle callbacks

Get notified when your custom element

- gets created
- is attached to the DOM
- is detached from the DOM
- has an attribute changed

# <my-element>

```
(function () {  
  var counter = 0,  
      element_prototype = Object.create(HTMLElement.prototype);  
  
  element_prototype.createdCallback = function () {  
    this.number = ++counter;  
    console.log('my-element[%d] created', this.number);  
  }  
  
  element_prototype.attachedCallback = function () {  
    this.textContent = '[my-element ' + this.number + ']';  
    console.log('my-element[%d] attached', this.number);  
  }  
})
```



# <my-element>

```
element_prototype.detachedCallback = function () {  
  console.log('my-element[%d] detached', this.number);  
}
```

```
element_prototype.attributeChangedCallback =  
  function (attr, oldVal, newVal) {  
    console.log('my-element[%d] attribute %s change: %s -> %s',  
      this.number, attr, oldVal, newVal);  
  }
```

# <my-element>

```
document.registerElement('my-element', {  
  prototype: element_prototype  
});
```

```
})();
```

# Demo: `<my-element>`

`demo/pre-load.html`

But what if `<my-element>` is defined  
*after* inserting it?

# Demo: `<my-element>` with delay

`demo/post-load.html`



This repository Search

Pull requests Issues Gist



github / time-elements

👁 Watch 43

★ Star 985

🍴 Fork 29

Web component extensions to the standard <time> element.

📦 220 commits

🌿 1 branch

📦 10 releases

👤 5 contributors



Branch: master ▾

time-elements / +



Merge pull request #45 from mateusortiz/master ...



Josh authored on Feb 18

latest commit 2e898a205c 📄

📁 examples	Update polyfill example	7 months ago
📁 test	Merge remote-tracking branch 'origin/master' into micro-time-ago	a year ago
📄 .gitignore	Add custom element dependencies	2 years ago
📄 .jshintrc	Tidy some linting	2 years ago
📄 .travis.yml	Make build step default	a year ago
📄 CONTRIBUTING.md	Update package name	a year ago
📄 LICENSE	Add LICENSE	2 years ago
📄 MAINTAINING.md	Update package name	a year ago
📄 Makefile	Make build step default	a year ago
📄 README.md	Update polyfill links	8 months ago
📄 bower.json	Time Elements 0.4.0	a year ago
📄 package.json	Upgrade packages	a year ago

<> Code

🔔 Issues 4

🔗 Pull requests 6

📶 Pulse

📊 Graphs

SSH clone URL

git@github.com:github

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

📄 Clone in Desktop

📄 Download ZIP

# Type Extensions

```
<time datetime="2015-02-18T15:20:33Z"  
  is="time-ago"  
  title="Feb 18, 2015, 4:20 PM GMT+1">7 months ago</time>
```

# Type Extensions

```
var TimeAgoPrototype = Object.create(HTMLTimeElement.prototype);  
  
window.TimeAgoElement = document.registerElement('time-ago', {  
  prototype: TimeAgoPrototype,  
  'extends': 'time'  
});
```



# Custom Elements - WD

Global48.14%

Denmark40.17%

Method of defining and using new types of DOM elements in a document.

Current alignedUsage relativeShow all

IE	Edge*	Firefox	Chrome	Safari	Opera	iOS Safari*	Opera Mini*	Android Browser*	Chrome for Android
8		38	31						
9		39	43					4.1	
10		40	44		31	7.1		4.3	
11	12	41	45	8	32	8.4		4.4.4	
	13	42	46	9	33	9	8	44	44
		43	47		34				
		44	48						


Notes

Known issues (0)

Resources (10)

Feedback

Current MS Edge status: Under Consideration

 Enabled through the "dom.webcomponents.enabled" preference in about:config

# Polyfill: webcomponents.js

<https://github.com/WebComponents/webcomponentsjs>

## Browser Support

Our polyfills are intended to work in the latest versions of evergreen browsers. See below for our complete browser support matrix:

Polyfill	IE10	IE11+	Chrome*	Firefox*	Safari 7+*	Chrome Android*	Mobile Safari*
Custom Elements	~	✓	✓	✓	✓	✓	✓
HTML Imports	~	✓	✓	✓	✓	✓	✓
Shadow DOM	✓	✓	✓	✓	✓	✓	✓
Templates	✓	✓	✓	✓	✓	✓	✓

**<math-tex>**

... but first:  
**<mathjax-loader>**

# mathjax-loader.js

```
(function () {  
  
    var states = {start: 1, loading: 2, ready: 3, typesetting: 4},  
        state = states.start,  
        queue = [],  
        src = 'https://cdn.mathjax.org/mathjax/latest/MathJax.js',  
        element_prototype = Object.create(HTMLElement.prototype);
```

# mathjax-loader.js

```
function load_mathjax(callback) {  
  state = states.loading;  
  window.MathJax = {  
    skipStartupTypeset: true,  
    jax: ['input/TeX', 'output/HTML-CSS'],  
    TeX: { extensions: ['AMSmath.js', 'AMSsymbols.js'] },  
    AuthorInit: function () {  
      MathJax.Hub.Register.StartupHook('End', callback);  
    }  
  };  
  var script = document.createElement('script');  
  script.type = 'text/javascript';  
  script.src = src;  
  script.async = true;  
  document.head.appendChild(script);  
}
```

# mathjax-loader.js

```
element_prototype.attachedCallback = function () {  
  if (this.hasAttribute('src'))  
    src = this.getAttribute('src');  
  load_mathjax(function () {  
    state = states.ready;  
    flush_queue()  
  });  
};
```

# mathjax-loader.js

```
function flush_queue() {  
  var to_process = queue.map(function (elem) {  
    return [MathJax.Hub.isJax(elem), elem];  
  }).filter(function (item) {  
    return item[0] !== 0;  
  });  
  queue = [];  
  if (to_process.length) {  
    state = states.typesetting;  
    to_process.forEach(function (item) {  
      var action = item[0] < 0 ? 'Typeset' : 'Reprocess';  
      MathJax.Hub.Queue([action, MathJax.Hub, item[1]]);  
    });  
    MathJax.Hub.Queue(flush_queue);  
  } else  
    state = states.ready;  
}
```



# mathjax-loader.js

```
element_prototype.typeset = function (elem) {  
  queue.push(elem);  
  if (state === states.ready)  
    flush_queue();  
};
```

# mathjax-loader.js

```
document.registerElement('mathjax-loader', {  
  prototype: element_prototype  
});
```

```
})();
```

... now back to  
<math-tex>

# math-tex.js

```
(function() {  
  
    var mathjax,  
        element_prototype = Object.create(HTMLElement.prototype);  
  
    function check_mathjax() {  
        if (mathjax) return;  
        mathjax = document.querySelector('mathjax-loader') ||  
            document.createElement('mathjax-loader');  
        if (!mathjax || typeof mathjax.typeset !== 'function')  
            console.warn('no mathjax-loader');  
        else if (!document.contains(mathjax))  
            document.head.appendChild(mathjax);  
    }  
})
```

# math-tex.js

```
element_prototype.createdCallback = function () {  
  check_mathjax();  
  this._jax = document.createElement('script');  
  this._jax.type = 'math/tex';  
};
```

# math-tex.js

```
element_prototype.attachedCallback = function () {  
  this._jax.text = this.textContent;  
  this.innerHTML = '';  
  this.appendChild(this._jax);  
  this.update();  
};
```

```
element_prototype.detachedCallback = function () {  
  this.textContent = this._jax.text;  
};
```

# math-tex.js

```
element_prototype.update = function () {  
  this._jax.type = this.getAttribute('display') === 'block' ?  
    'math/tex; mode=display' : 'math/tex';  
  if (mathjax && this._jax.parentNode === this)  
    mathjax.typeset(this._jax);  
}
```

# math-tex.js

```
element_prototype.attributeChangedCallback = function (attr) {  
  if (attr === 'display')  
    this.update();  
};
```



# math-tex.js

```
Object.defineProperty(element_prototype, 'math', {  
  get: function () {  
    return this._jax.text;  
  },  
  set: function (value) {  
    this._jax.text = value;  
    this.update();  
  }  
});
```

# math-tex.js

```
document.registerElement('math-tex', {  
  prototype: element_prototype  
});
```

```
})();
```

# Demo: Dynamic math again

[demo/dynamic-math.html](#)

# We can do better

- ShadowDOM
- MutationObserver
- Custom events

# Demo: A better $\text{<math>tex\text{>}$

[demo/better-math-tex.html](#)

# Some resources

- <http://webcomponents.org>
- <https://customelements.io>
- <https://github.com/janmarthedal/math-tex>

# Some resources

- <http://webcomponents.org>
- <https://customelements.io>
- <https://github.com/janmarthedal/math-tex>

# Thank you!