sheffercool / **tribute**

forked from zurb/tribute

ES6 Native @mentions

zurb.github.io/tribute/example/

MIT License

0 stars 199 forks

Star | Watch

Code | Pull requests | Actions | Projects | Security | Insights | Settings

master

This branch is even with zurb:master.

Pull request ± Compare

cgross and **jseminck** … on 7 Jun

View code

README.md

Tribute

cdnjs v5.1.3 build passing

A cross-browser `@mention` engine written in ES6, no dependencies. Tested in Firefox, Chrome, iOS Safari, Safari, IE 9+, Edge 12+, Android 4+, and Windows Phone.

- Example

Installing

There are a few ways to install Tribute; Bower, as an NPM Module, or by downloading from the `dist` folder in this repo.

NPM Module

You can install Tribute by running:

```
npm install tributejs
```

Or by adding Tribute to your `package.json` file.

Import into your ES6 code.

```
import Tribute from "tributejs";
```

Ruby Gem

To use Tribute within a Rails project, you can add the following to the app's Gemfile:

```
gem 'tribute'
```

Then, add the following to `app/assets/javascripts/application.js` :

```
*= require tribute
```

And in `app/assets/stylesheets/application.css` :

```
//= require tribute
```

Webpack

To add Tribute to your webpack build process, start by adding it to your package.json and running `npm install` .

After installing, you need to update your Babel module loader to not exclude Tribute from being compiled by Webpack:

```
{
    test: /\.js$/,
    loader: 'babel',
    exclude: /node_modules\/(?!tributejs)/
}
```

Download or Clone

Or you can download the repo or clone it localy with this command:

```
git clone git@github.com:zurb/tribute.git
```

You can then copy the files in the `dist` directory to your project.

```
<link rel="stylesheet" href="js/tribute.css" />
<script src="js/tribute.js"></script>
```

That's it! Now you are ready to initialize Tribute.

Initializing

There are two ways to initialize Tribute, by passing an array of "collections" or by passing one collection object.

```
var tribute = new Tribute({
  values: [
    { key: "Phil Heartman", value: "pheartman" },
    { key: "Gordon Ramsey", value: "gramsey" }
  ]
});
```

You can pass multiple collections on initialization by passing in an array of collection objects to `collection`.

```
var tribute = new Tribute({
  collection: []
});
```

Attaching to elements

Once initialized, Tribute can be attached to an `input`, `textarea`, or an element that supports `contenteditable`.

```
<div id="caaanDo">I'm Mr. Meeseeks, look at me!</div>

<div class="mentionable">Some text here.</div>
<div class="mentionable">Some more text over here.</div>

<script>
  tribute.attach(document.getElementById("caaanDo"));

  // also works with NodeList
  tribute.attach(document.querySelectorAll(".mentionable"));
</script>
```

A Collection

Collections are configuration objects for Tribute, you can have multiple for each instance. This is useful for scenarios where you may want to match multiple trigger keys, such as `@` for users and `#` for projects.

Collection object shown with defaults:

```
{
  // symbol or string that starts the lookup
  trigger: '@',

  // element to target for @mentions
  iframe: null,

  // class added in the flyout menu for active item
  selectClass: 'highlight',

  // class added to the menu container
  containerClass: 'tribute-container',

  // class added to each list item
  itemClass: '',

  // function called on select that returns the content to insert
  selectTemplate: function (item) {
    return '@' + item.original.value;
  },

  // template for displaying item in menu
  menuItemTemplate: function (item) {
    return item.string;
  },

  // template for when no match is found (optional),
  // If no template is provided, menu is hidden.
  noMatchTemplate: null,
  // specify an alternative parent container for the menu
  // container must be a positioned element for the menu to appear correctly ie. `pos
  // default container is the body
  menuContainer: document.body,

  // column to search against in the object (accepts function or string)
  lookup: 'key',

  // column that contains the content to insert by default
  fillAttr: 'value',

  // REQUIRED: array of objects to match or a function that returns data (see 'Loadin
  values: [],

  // When your values function is async, an optional loading template to show
  loadingItemTemplate: null,

  // specify whether a space is required before the trigger string
  requireLeadingSpace: true,

  // specify whether a space is allowed in the middle of mentions
  allowSpaces: false,

  // optionally specify a custom suffix for the replace text
  // (defaults to empty space if undefined)
  replaceTextSuffix: '\n',

  // specify whether the menu should be positioned.  Set to false and use in conjucti
  // (defaults to true)
  positionMenu: true,

  // when the spacebar is hit, select the current match
  spaceSelectsMatch: false,

  // turn tribute into an autocomplete
  autocompleteMode: false,

  // Customize the elements used to wrap matched strings within the results list
  // defaults to <span></span> if undefined
  searchOpts: {
    pre: '<span>',
    post: '</span>',
    skip: false // true will skip local search, useful if doing server-side search
  },

  // Limits the number of items in the menu
  menuItemLimit: 25,

  // specify the minimum number of characters that must be typed before menu appears
  menuShowMinLength: 0
}
```

Dynamic lookup column

The `lookup` column can also be passed a function to construct a string to query against. This is useful if your payload has multiple attributes that you would like to query against but you can't modify the payload returned from the server to include a concatenated lookup column.

```
{
  lookup: function (person, mentionText) {
    return person.name + person.email;
  }
}
```

Template Item

Both the `selectTemplate` and the `menuItemTemplate` have access to the `item` object. This is a meta object containing the matched object from your values collection, wrapped in a search result.

```
{
  index: 0;
  original: {
  } // your original object from values array
  score: 5;
  string: "<span>J</span><span>o</span>rdan Hum<span>p</span>hreys";
}
```

Trigger tribute programmatically

Tribute can be manually triggered by calling an instances `showMenuForCollection` method. This is great for trigging tribute on an input by clicking an anchor or button element.

```
<a id="activateInput">@mention</a>
```

Then you can bind a `mousedown` event to the anchor and call `showMenuForCollection` .

```
activateLink.addEventListener("mousedown", function(e) {
  e.preventDefault();
  var input = document.getElementById("test");

  tribute.showMenuForCollection(input);
});
```

Note that `showMenuForCollection` has an optional second parameter called `collectionIndex` that defaults to 0. This allows you to specify which collection you want to trigger with the first index starting at 0.

For example, if you want to trigger the second collection you would use the following snippet: `tribute.showMenuForCollection(input, 1);`

Events

Replaced

You can bind to the `tribute-replaced` event to know when we have updated your targeted Tribute element.

If your element has an ID of `myElement` :

```
document
  .getElementById("myElement")
  .addEventListener("tribute-replaced", function(e) {
    console.log(
      "Original event that triggered text replacement:",
      e.detail.event
    );
    console.log("Matched item:", e.detail.item);
  });
```

No Match

You can bind to the `tribute-no-match` event to know when no match is found in your collection.

If your element has an ID of `myElement` :

```
document
  .getElementById("myElement")
  .addEventListener("tribute-no-match", function(e) {
    console.log("No match found!");
  });
```

Active State Detection

You can bind to the `tribute-active-true` or `tribute-active-false` events to detect when the menu is open or closed respectively.

```
document
  .getElementById("myElement")
  .addEventListener("tribute-active-true", function(e) {
    console.log("Menu opened!");
  });
```

```
document
  .getElementById("myElement")
  .addEventListener("tribute-active-false", function(e) {
    console.log("Menu closed!");
  });
```

Tips

Some useful approaches to common roadblocks when implementing @mentions.

Updating a collection with new data

You can update an instance of Tribute on the fly. If you have new data you want to insert into the current active collection you can access the collection values array directly:

```
tribute.appendCurrent([
  { name: "Howard Johnson", occupation: "Panda Wrangler", age: 27 },
  { name: "Fluffy Croutons", occupation: "Crouton Fluffer", age: 32 }
]);
```

This would update the first configuration object in the collection array with new values. You can access and update any attribute on the collection in this way.

You can also append new values to an arbitrary collection by passing an index to `append` .

```
tribute.append(2, [
  { name: "Howard Johnson", occupation: "Panda Wrangler", age: 27 },
  { name: "Fluffy Croutons", occupation: "Crouton Fluffer", age: 32 }
]);
```

This will append the new values to the third collection.

Programmatically detecting an active Tribute dropdown

If you need to know when Tribute is active you can access the `isActive` property of an instance.

```
if (tribute.isActive) {
  console.log("Somebody is being mentioned!");
} else {
  console.log("Who's this guy talking to?");
}
```

Links inside contenteditable are not clickable.

If you want to embed a link in your `selectTemplate` then you need to make sure that the anchor is wrapped in an element with `contenteditable="false"` . This makes the anchor clickable *and* fixes issues with matches being modifiable.

```
var tribute = new Tribute({
  values: [
    {
      key: "Jordan Humphreys",
      value: "Jordan Humphreys",
      email: "getstarted@zurb.com"
    },
    {
      key: "Sir Walter Riley",
      value: "Sir Walter Riley",
      email: "getstarted+riley@zurb.com"
    }
  ],
  selectTemplate: function(item) {
    return (
      '<span contenteditable="false"><a href="http://zurb.com" target="_blank" title=
      item.original.email +
      '">' +
      item.original.value +
      "</a></span>"
    );
  }
});
```

How do I add an image to the items in the list?

You can override the default `menuItemTemplate` with your own output on initialization. This allows you to replace the `innerHTML` of the `li` of each item in the list. You can use `item.string` to return the markup for the fuzzy match.

```
{
  //..other config options
  menuItemTemplate: function (item) {
    return '<img src="'+item.original.avatar_url + '">' + item.string;
    }
  }
```

Embedding Tribute in a scrollable container.

Sometimes you may need to have the Tribute menu attach to a scrollable parent element so that if the user scrolls the container the menu will scroll with it. To do this, you can set `menuContainer` to the node that is the scrollable parent.

```
{
  //..other config options
  menuContainer: document.getElementById("wrapper");
}
```

Loading remote data

If your data set is large or would like to pre filter your data you can load dynamically by setting the `values` to a function.

```
{
  //..other config options
  // function retrieving an array of objects
  values: function (text, cb) {
    remoteSearch(text, users => cb(users));
  },
  lookup: 'name',
  fillAttr: 'name'
}
```

You would then define a function, in this case `remoteSearch`, that returns your data from the backend.

```
function remoteSearch(text, cb) {
  var URL = "YOUR DATA ENDPOINT";
  xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
      if (xhr.status === 200) {
        var data = JSON.parse(xhr.responseText);
        cb(data);
      } else if (xhr.status === 403) {
        cb([]);
      }
    }
  };
  xhr.open("GET", URL + "?q=" + text, true);
```

```
    xhr.send();
  }
```

Hide menu when no match is returned

If you want the menu to not show when no match is found, you can set your `noMatchTemplate` config to the following:

```
noMatchTemplate: function () {
  return '<span style:"visibility: hidden;"></span>';
}
```

Detaching Tribute instances

When you want to remove Tribute from an element you can call `detach`.

```
tribute.detach(document.getElementById("caaanDo"));
```

This will remove all event listeners from the DOM that are associated with that element.

Trigger on multiple character strings

It is also possible to configure Tribute to trigger on a string consisting of multiple characters.

This example shows the usage of Tribute for autocompletion of variables:

```
var tribute = new Tribute({
  trigger: "{{",
  values: [
    { key: "red", value: "#FF0000" },
    { key: "green", value: "#00FF00" }
  ],
  selectTemplate: function(item) {
    return "{{" + item.original.key + "}}";
  },
  menuItemTemplate: function(item) {
    return item.original.key + " = " + item.original.value;
  }
});
```

Framework Support

Vue.js — vue-tribute by **@syropian**

AngularJS 1.5+ — angular-tribute by **ZURB**

Angular 2+ - ngx-tribute by **Ladder.io**

Ruby — tribute-rb by **ZURB**

Ember – ember-tribute by **MalayaliRobz**

WYSIWYG Editor Support

- Froala Editor - https://www.froala.com/wysiwyg-editor/examples/tribute-js

Brought to you by

ZURB, the creators of Helio

Design successful products by rapidly revealing key user behaviors. Helio makes it easy to get reactions on your designs quickly so your team can focus on solving the right problems, right now.

Releases

62 tags

Create a new release

Packages

No packages published
Publish your first package

Languages

● **JavaScript** 91.4% ● **HTML** 7.8% ● **CSS** 0.8%