Sharing is Caring!

Patterns for JavaScript Library Design

@maggiepint

Who am I?

Who am I?

- Semicolons/Tabs

Who am I?

- Crisis Management Engineer at Microsoft
- A maintainer (not the author) of Moment.js
- JS Foundation Representative to TC39
- Champion of Date rework in JavaScript (tell me your thoughts!)

Twitter: @maggiepint

Email: maggiepint@gmail.com

A library is a bit of code that is useful when packaged up and distributed to other people.

This could be internal or external.

Libraries

1. LoDash
2. jQuery
3. Q
4. Moment
5. Immutable
6. Request

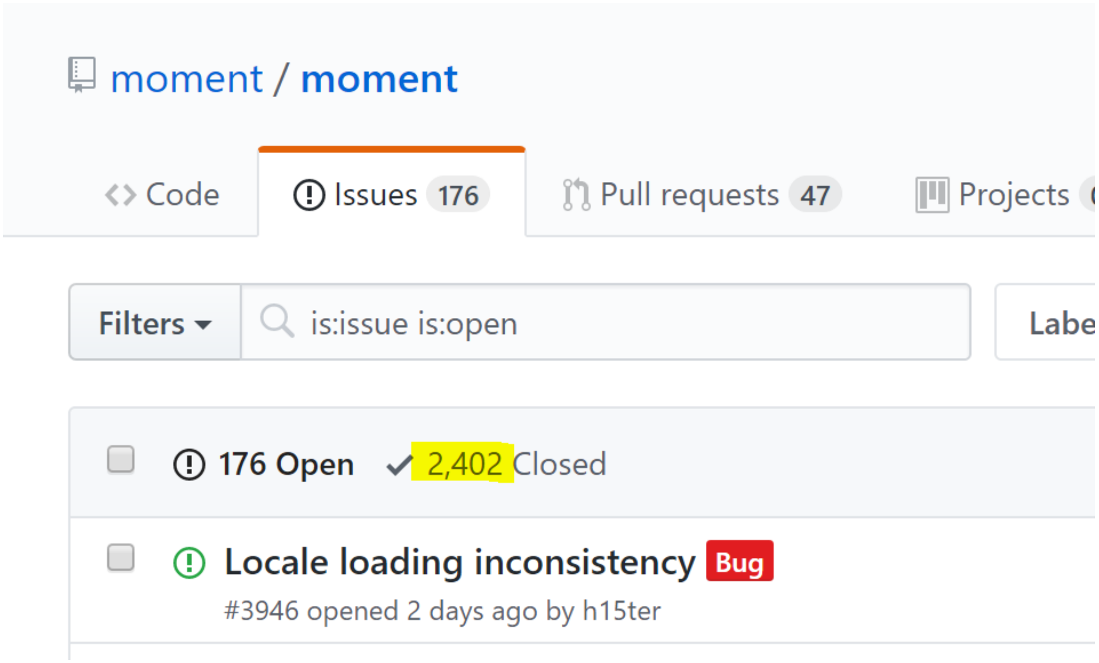
Not Libraries

1. Express
2. Angular
3. Webpack

What we think having a library is like

What having a library is actually like

geek



moment / moment

Code | Issues 176 | Pull requests 47 | Projects

Filters | is:issue is:open | Labe

176 Open ✓ 2,402 Closed

Locale loading inconsistency Bug
#3946 opened 2 days ago by h15ter

What makes a library good?

What makes a library good?

- Small Size!

What makes a library good?

- ~~Small Size!~~

What makes a library good?

- ~~Small Size!~~
- Great Code!

What makes a library good?

- ~~Small Size!~~
- ~~Great Code!~~

What makes a library good?

- ~~Small Size!~~
- ~~Great Code!~~
- Encourages functional programming practices

What makes a library good?

- ~~Small Size!~~
- ~~Great Code!~~
- ~~Encourages functional programming practices~~

What makes a library good?

- ~~Small Size!~~
- ~~Great Code!~~
- ~~Encourages functional programming practices~~
- Amazing nodejs/webpack/babel/mocha/chai/phantom/sauce/istanbul toolchain!

What makes a library good?

- ~~Small Size!~~
- ~~Great Code!~~
- ~~Encourages functional programming practices~~
- ~~Amazing nodejs/webpack/babel/mocha/chai/phantom/sauce/istanbul toolchain!~~

What makes a library good?

- ~~Small Size!~~
- ~~Great Code!~~
- ~~Encourages functional programming practices~~
- ~~Amazing nodejs/webpack/babel/mocha/chai/phantom/sauce/istanbul toolchain!~~
- Ease of use

Ease of Use!

Nobody wants to learn your library.

It's okay to make it simple.

Considerations

1. Invocation
2. Configuration
3. Defaults
4. Errors

Invocation

Static Invocation

(Simplest Option)

```
// request
request('http://www.google.com', (err, resp, body) => {
});
```

```
// lodash
let odds = _.filter([1,2,3,4,5], (n) => { return n % 2});
```

The drawback:

```
let arr = [1,2,3,4,5];
let sumOddsDoubled = _sum(
                          _.map(
                              _.filter(arr, (n) => { return n % 2})
                                 , (n) => {return n *2 }));
```

Factory Function

(Extends well)

```
// Q
let promise = Q.fcall();
let promise = Q.all();
let deferred = Q.defer();

// jQuery
let obj = $('.blue');

//moment
let myTime = moment();
let myTime = moment.utc();
```

Chaining

(AKA Fluent)

```
Q.fcall(promisedStep1)
.then(promisedStep2)
.then(promisedStep3)
.then(promisedStep4)
.then(function (value4) {
    // Do something with value4
})
.catch(function (error) {
    // Handle any error from all above steps
})
.done();
```

```
moment().add(3, 'days').startOf('day').subtract(1, 'year');
```

```
$('.blue').append('<p>these elements have a blue class</p>')
    .class('emphasize);
```

Configuration

First there was:

```
let dateString = 'May 1, 2017';
moment(dateString);
moment(dateString, 'MMM D, YYYY');
```

First there was:

```
let dateString = 'May 1, 2017';
moment(dateString);
moment(dateString, 'MMM D, YYYY');
```

And then there was:

```
moment([2017,2,11]);
moment(moment());
moment(new Date());
moment(dateString, 'MMM D, YYYY', 'en');
moment(dateString, 'MMM D, YYYY', true);
moment(dateString, 'MMM D, YYYY', 'en', true);
moment(dateString, ['MMM D, YYYY', 'YYYY-MM-DD']);
moment(dateString, ['MMM D, YYYY', 'YYYY-MM-DD'],'en');
moment(dateString, ['MMM D, YYYY', 'YYYY-MM-DD'], true);
moment(dateString, ['MMM D, YYYY', 'YYYY-MM-DD'],'en', true);
```

Options Objects

(The ECMA402 Intl API)

```
let options = {
  hour: 'numeric',
  minute: 'numeric',
  second: 'numeric',
  timeZoneName: 'short'
};

let dateString = new Intl.DateTimeFormat('en-AU', options).format(date);
```

Options allow for pluggable business logic

```javascript
let options = {
  statusCode: {
    404: function() {
      alert( "page not found" );
    },
    500: function() {
        throw 'the world has ended';
    }
  }
};

$.ajax(options);
```

Required paramaters first, then an options object

Defaults

Plain HTTP Request with Node.JS

```
let http = require('http');
http.get({
        host: 'api.github.com',
        path: '/repos/timrwood/moment',
        headers: {
    'User-Agent': 'request'
  }
    }, function(response) {
        // Continuously update stream with data
        let body = '';
        response.on('data', function(d) {
            body += d;
        });
        response.on('end', function() {
              console.log(response.statusCode); // 301
              console.log(response.statusMessage); // Moved Permanently
        });
    });
```

Sensible Defaults with Request

```javascript
let request = require('request');

let options = {
  url: 'https://api.github.com/repos/timrwood/moment',
  headers: {
    'User-Agent': 'request'
  }
};
request(options, (err, resp, body) => {
    console.log(resp.request.uri.href);
    // https://api.github.com/repositories/1424470
    console.log(resp.body);
    // {"id":1424470,"name":"moment","full_name":"moment/moment", ...
});
```

Best By Default

If there is an obvious right answer, do it automatically.

```
let dateTimeString = '2017-05-01 10:25:41.357-05:00';
moment(dateTimeString).format('LLL');
```

1 May 2017 17:25

WAT?

```javascript
let dateTimeString = '2017-05-01 10:25:41.357-05:00';
moment(dateTimeString).format('LLL');
// 1 May 2017 17:25
moment.utc(dateTimeString).format('LLL');
// 1 May 2017 15:25
moment.parseZone(dateTimeString).format('LLL');
//  1 May 2017 10:25
moment.tz(dateTimeString, 'America/New_York').format('LLL');
// 1 May 2017 11:25
```

There is no best choice!

```
let dateTimeString = '2017-05-01 10:25:41.357-05:00';
moment.local(dateTimeString).format('LLL');
```

1 May 2017 17:25

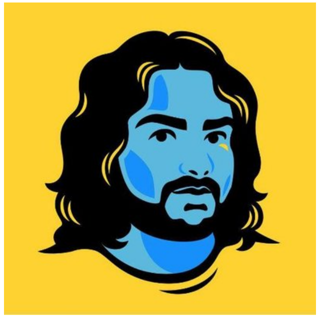
Deafult When Best

If there are several likely behaviors, don't choose a deafult. It only leads to confusion.

Errors

> *Any use of the JavaScript throw mechanism will raise an exception that must be handled using try / catch or the Node.js process will exit immediately.*

The Node.js Docs



Myles cares a lot about errors.

```
moment('asdfgh', 'DD/MM/YYYY').format() // invalid date
moment('asdfgh', 'DD/MM/YYYY').format() // invalid date
```

Good! Bad user input doesn't crash the Node.js process.

```
moment().get('hours'); // 13
moment().get('hours'); // 13
```

```
moment().get('huors');
// Moment {_isAMomentObject: true, _isUTC: false, _pf: ...
```

```
moment().get('hours'); // 13
```

```
moment().get('huors');
// Moment {_isAMomentObject: true, _isUTC: false, _pf: ...
```

¯\(ツ)/¯

Throw on Obvious Developer Error

```
const { Map } = require('immutable');
const map1 = Map({ a: 1, b: 2, c: 3 });
const map2 = map1.set('b', 50)
let b1 = map1.get('b'); //2
let b2 = map2.get('b')); // 50
```

```
const { Map } = require('immutable');
const map3 = Map(1);
// TypeError: Expected Array or iterable object of [k, v] entries,
// or keyed object: 1
```

Make it easy to use!

Invocation

- Start with basic static or factory invocation
- Consider a chaining API

Configuration

- Use options objects for non-required configurations

Defaults

- Best-by-default
- Deafult-when-best

Errors

- Throw for obvious developer errors

Share!

@maggiepint - maggiepint@gmail.com - https://www.maggiepint.com