
Basil.js

The missing Javascript smart persistence layer. Unified localStorage, cookie and session storage JavaScript API.

Philosophy

Basil aims to ease the frontend storage management for developers. It strives to be bulletproof and handle disabled cookies, full localStorage and other unwanted native storage exceptions..

When you try to store something, basil will automatically look through all the available storage mechanisms and find the best suited one to store your value. It also handles storage of complex javascript objects using json.

Basic Usage

```
1 basil = new window.Basil(options);
2
3 // basic methods
4 basil.set('foo', 'bar'); // store 'bar' value under 'foo' key
5 basil.set('abc', 'xyz'); // store 'xyz' value under 'abc' key
6 basil.get('foo'); // returns 'bar'
7 basil.keys(); // returns ['abc', 'foo']
8 basil.keysMap(); // returns { 'abc': ['local'], 'foo': ['local'] }
9 basil.remove('foo'); // remove 'foo' value
10
11 // advanced methods
12 basil.check('local'); // boolean. Test if localStorage is available
13 basil.reset(); // reset all stored values under namespace
```

Advanced Usage

Storages

```
1 basil = new window.Basil(options);
2
3 // force storage on the go through basil
4 // set 'bar' value under 'foo' key in localStorage
5 basil.set('foo', 'bar', { 'storages': ['local'] });
6
7 // set 'bar' value under 'foo' key.
8 // try first to store it into cookies and if not possible into
  localStorage
```

```

 9 basil.set('foo', 'quux', { 'storages': ['cookie', 'local'] });
10
11 // set 'xyz' value under 'abc' key in memory
12 basil.set('abc', 'xyz', { 'storages': ['memory'] });
13
14 // set value without JSON encoding
15 basil.set('foo', '{ "bar": "baz" }', { raw: true }); // will save { "
    bar": "baz" } as string
16
17 // retrieve keys
18 basil.keys(); // returns ['foo', 'abc']
19 basil.keys({ 'storages': ['memory'] }); // returns ['abc']
20
21 // retrieve keys map
22 basil.keysMap(); // returns { 'foo': ['local', 'cookie'], 'abc': ['
    memory'] }
23 basil.keysMap({ 'storages': ['memory'] }); // returns { 'abc': ['memory
    '] }
```

Native storages

```

 1 // Access native storages
 2 // With basil API, but without namespace nor JSON parsing for values
 3
 4 // cookies has specific options
 5 Basil.cookie.get(key);
 6 Basil.cookie.set(key, value, {
 7     'expireDays': days,
 8     'domain': 'mydomain.com',
 9     'secure': true,
10     'sameSite': 'strict'
11 });
12
13 // localStorage
14 Basil.localStorage.get(key);
15 Basil.localStorage.set(key, value);
16
17 // sessionStorage
18 Basil.sessionStorage.get(key);
19 Basil.sessionStorage.set(key, value);
```

Namespaces

```

 1 basil = new window.Basil(options);
 2
 3 // store data under default namespace
 4 basil.set('hello', 'world');
```

```
5
6 // store data under a given namespace
7 basil.set('hello', 42, { 'namespace': 'alt' });
8 basil.set('abc', 'def', { 'namespace': 'alt', 'storages': ['memory'] })
9
10 // retrieve data
11 basil.get('hello'); // return 'world'
12 basil.get('hello', { 'namespace': 'alt' }); // return 42
13
14 // retrieves keys
15 basil.keys(); // returns ['hello']
16 basil.keys({ 'namespace': 'alt' }); // returns ['hello', 'abc']
17
18 // retrieves keys map
19 basil.keysMap(); // returns { 'hello': ['local'] }
20 basil.keysMap({ 'namespace': 'alt' }); // returns { 'hello': ['local'],
    'abc': ['memory'] }
21
22 // remove data under a given namespace
23 basil.remove('hello', { 'namespace': 'alt' });
24 basil.get('hello'); // return 'world'
25 basil.get('hello', { 'namespace': 'alt' }); // return null
26
27 // reset data under a given namespace
28 basil.reset({ 'namespace': 'alt', 'storages': ['local', 'memory']});
```

Configuration

Here is the whole `options` object that you could give to Basil:

```
1 options = {
2   // Namespace. Namespace your Basil stored data
3   // default: 'b45i1'
4   namespace: 'foo',
5
6   // storages. Specify all Basil supported storages and priority order
7   // default: `['local', 'cookie', 'session', 'memory']`
8   storages: ['cookie', 'local']
9
10  // expireDays. Default number of days before cookies expiration
11  // default: 365
12  expireDays: 31
13
14  // keyDelimiter. The value used delimit the namespace from the key
    name
15  // default: '.'
16  keyDelimiter: '.'
17 }
```

```
18 };
```

Compatibility

- Firefox 3.5+
- Internet Explorer 7 (requires json2.js)
- Internet Explorer 8+
- Chrome 4+
- Safari 4+

Plugins

List plugin

This plugin mimics Redis Lists methods and behaviors. Here are the (yet) supported methods.

```
1 basil = new window.Basil(options);
2 basil.lindex(key, index);
3 basil.linsert(key, where, pivot, value);
4 basil.llen(key);
5 basil.lpop(key);
6 basil.lpush(key, value);
7 basil.lrange(key, start, stop);
8 basil.lrem(key, count, value);
9 basil.lset(key, index, value);
10 basil.ltrim(key, start, stop);
11 basil.rpop(key);
12 basil.rpush(key, value);
```

Set plugin

This plugin mimics Redis Sets methods and behaviors. Except sscan all the methods are implemented.

```
1 basil = new window.Basil(options);
2 basil.sadd(key, member [members ...]);
3 basil.scard(key);
4 basil.sdiff(key [keys ...]);
5 basil.sdiffstore(destination, key [keys ...]);
6 basil.sinter(key [keys ...]);
7 basil.sinterstore(destination, key [keys ...]);
8 basil.sismember(key, member);
9 basil.smembers(key);
```

```
10 basil.smove(source, destination, member);
11 basil.spop(key);
12 basil.srandmember(key, [count]);
13 basil.srem(key, member [members ...]);
14 basil.sunion(key [keys ...]);
15 basil.sunionstore(destination, key [keys ...]);
```

Build

To generate the production files, make sure you have already installed the dependencies using `npm install` and then just use:

```
1 npm run-script build
```

Tests

To launch the test suite, make sure you have already installed the dependencies using `npm install`. Tests are launching in all your installed browsers. They're also launched on Travis CI, in PhantomJS.

```
1 npm test
2 npm run-script test-plugins
```

License

MIT. See `LICENSE.md`