Zone.js's support for non standard apis

Zone.js patched most standard APIs such as DOM event listeners, XMLHttpRequest in Browser, EventEmitter and fs API in Node.js so they can be in zone.

But there are still a lot of non-standard APIs that are not patched by default, such as MediaQuery, Notification, WebAudio and so on. This page provides updates on the current state of zone support for Angular APIs.

Currently supported non-standard Web APIs

- MediaQuery
- Notification

Currently supported polyfills

- webcomponents

Usage:

```
<script src="webcomponents-lite.js"></script>
<script src="node_modules/zone.js/bundles/zone.umd.js"></script>
<script src="node_modules/zone.js/bundles/webapis-shadydom.umd.js"></script>
```

Currently supported non standard node APIs

Currently supported non standard common APIs

- Bluebird Promise

Browser Usage:

```
  <script src="zone.js"></script>
  <script src="bluebird.js"></script>
  <script src="zone-bluebird.js"></script>
  <script>
  Zone[Zone['__symbol__']('bluebird')](Promise);
  </script>
```

After those steps, window.Promise becomes Bluebird Promise and will also be zone awareness.

Angular Usage:

in polyfills.ts, import the `zone-bluebird` package.

```
import 'zone.js'; // Included with Angular CLI.
import 'zone.js/plugins/zone-bluebird';
```

in main.ts, patch bluebird.

```
platformBrowserDynamic()
    .bootstrapModule(AppModule)
    .then(_ => {
import('bluebird').then(Bluebird => {const Zone = (window as any)['Zone'];
```

```
Zone[Zone['__symbol__']('bluebird')](Bluebird.default);});
    })
    .catch(err => console.error(err));
```

After this step, the `window.Promise` will be the Bluebird `Promise`, and the callback of `Bluebird.then` will be executed in the Angular zone.

Node Sample Usage:

```
require('zone.js');
const Bluebird = require('bluebird');
require('zone.js/plugins/zone-bluebird');
Zone[Zone['__symbol__']('bluebird')](Bluebird);
Zone.current.fork({
  name: 'bluebird'
}).run(() => {
  Bluebird.resolve(1).then(r => {
    console.log('result ', r, 'Zone', Zone.current.name);
  });
});
```

In NodeJS environment, you can choose to use Bluebird Promise as global.Promise or use ZoneAwarePromise as global.Promise.

To run the jasmine test cases of bluebird

```
  npm install bluebird
```

then modify test/node_tests.ts remove the comment of the following line

```
//import './extra/bluebird.spec';
```

Others

- Cordova

patch `cordova.exec` API

```
cordova.exec(success, error, service, action, args);
```

`success` and `error` will be patched with `Zone.wrap`.

to load the patch, you should load in the following order.

```
<script src="zone.js"></script>
<script src="cordova.js"></script>
<script src="zone-patch-cordova.js"></script>
```

Usage

By default, those APIs' support will not be loaded in zone.js or zone-node.js, so if you want to load those API's support, you should load those files by yourself.

For example, if you want to add MediaQuery patch, you should do like this:

```
  <script src="path/zone.js"></script>
  <script src="path/webapis-media-query.js"></script>
```

- rxjs

`zone.js` also provide a `rxjs` patch to make sure rxjs Observable/Subscription/Operator run in correct zone. For details please refer to pull request 843. The following sample code describes the idea.

```
const constructorZone = Zone.current.fork({name: 'constructor'});
const subscriptionZone = Zone.current.fork({name: 'subscription'});
const operatorZone = Zone.current.fork({name: 'operator'});

let observable;
let subscriber;
constructorZone.run(() => {
  observable = new Observable((_subscriber) => {
    subscriber = _subscriber;
    console.log('current zone when construct observable:', Zone.current.name); // will
output constructor.
    return () => {
      console.log('current zone when unsubscribe observable:', Zone.current.name); //
will output constructor.
    }
  });
});

subscriptionZone.run(() => {
  observable.subscribe(() => {
    console.log('current zone when subscription next', Zone.current.name); // will
output subscription.
  }, () => {
    console.log('current zone when subscription error', Zone.current.name); // will
output subscription.
  }, () => {
    console.log('current zone when subscription complete', Zone.current.name); // will
output subscription.
  });
});

operatorZone.run(() => {
  observable.map(() => {
    console.log('current zone when map operator', Zone.current.name); // will output
operator.
  });
});
```

Currently basically everything the `rxjs` API includes

- Observable
- Subscription

- Subscriber
- Operators
- Scheduler

is patched, so each asynchronous call will run in the correct zone.

Usage.

For example, in an Angular application, you can load this patch in your `app.module.ts`.

```
import 'zone.js/plugins/zone-patch-rxjs';
```

- electron

In electron, we patched the following APIs with `zone.js`

1. Browser API
2. NodeJS
3. Electorn Native API

Usage.

add following line into `polyfill.ts` after loading zone-mix.

```
//import 'zone.js'; // originally added by angular-cli, comment it out
import 'zone.js/mix'; // add zone-mix to patch both Browser and Nodejs
import 'zone.js/plugins/zone-patch-electron'; // add zone-patch-electron to patch
Electron native API
```

there is a sampel repo zone-electron.

- socket.io-client

user need to patch `io` themselves just like following code.

```
    <script src="socket.io-client/dist/socket.io.js"></script>
    <script src="zone.js/bundles/zone.umd.js"></script>
    <script src="zone.js/bundles/zone-patch-socket-io.js"></script>
    <script>
      // patch io here
      Zone[Zone.__symbol__('socketio')](io);
    </script>
```

please reference the sample repo zone-socketio about detail usage.

- jsonp

Usage.

provide a helper method to patch jsonp. Because jsonp has a lot of implementation, so user need to provide the information to let json `send` and `callback` in zone.

there is a sampel repo zone-jsonp here, sample usage is:

```
import 'zone.js/plugins/zone-patch-jsonp';
Zone['__zone_symbol__jsonp']({
  jsonp: getJSONP,
  sendFuncName: 'send',
  successFuncName: 'jsonpSuccessCallback',
  failedFuncName: 'jsonpFailedCallback'
});
```

- ResizeObserver

Currently only `Chrome 64` native support this feature. you can add the following line into `polyfill.ts` after loading `zone.js`.

```
import 'zone.js';
import 'zone.js/plugins/zone-patch-resize-observer';
```

there is a sample repo zone-resize-observer here