Runtime-Core

- Release 0.1.0
- Example
- Tasks
- Notes

Setup Environment

On the first time you are cloning this repository, you need to run the command:

```
npm run init-setup
```

After running successfully this command you will have 2 folders (node_modules and vendor), these folders are excluded from the commit process, and are only for development.

if you already have the project configured on your machine, you only need run the next command to add new dependencies:

```
npm install
jspm install
```

Private Repository Note

if you have problems with the `npm install` command, like "access was forbidden", "404 not found", and have the service framework module reference, it is an authentication problem;

you may need following the steps present on Github Help. and select operation system you are using.

This could happen because it is a private module and need your GitHub authentication to allow cloning the repository.

If you have some troubles with authentication on windows using the Git Shell, you can try caching your GitHub password. This should avoid the constant prompt for username and password;

Instalation through jspm

We need configure jspm config using github tokens, for that, following this (based on issue 3):

1. Here, generate token with public_repo permission enabled
2. Save the token generated;
3. Execute the command `jspm registry config github` and you'll be asked for the credentials;
4. Now you can execute command `jspm install -y` and the `runtime-core` or `jspm install runtime-core=github:reTHINK-project/dev-runtime-core` or only

```
    jspm install;
```

Issues

if you have some trouble with the environment, you can open an issue;

Javascript Environment

JavaScript code should be written in ES6. There are direct dependencies from nodejs and npm, these can be installed separately or in conjunction with nvm

Dependencies

- nodejs
- npm
- karma - Make the communication between unit test tool and jenkins. See more on karma
- mocha - Unit test tool. See more on http://mochajs.org
- jspm - Don't need compile the code, it uses babel (or traucer or typescript) to run ES6 code on browser. Know more in jspm.io
- gulp - Automate and enhance your workflow. See more about gulp on gulp

Code Style and Hinting

On the root directory you will find **.jshintrc** and **.jscsrc**, these files are helpers to maintain syntax consistency, it signals syntax mistakes and makes the code equal for all developers.

- jscs - Maintain JavaScript Code Style
- jshint - Detect errors and potential problems in JavaScript code.

All IDE's and Text Editors can handle these tools.

Documentation

To generates api documentation you can run `gulp doc`

Unit Testing

Unit testing can be launched manually with **karma start**.

~~It's advisable to use expect.js instead of assert.~~

After investigate and testing the expect.js it don't support some features for ES6, because this tool hasn't activity at some time, that is why, it is recomended use the chaijs it is more versatile and have expect.js (but updated) and others tools that can be useful;

How to include this runtime-core code into others parts of reTHINK Project;

How to include this repository in other software parts, like dev-runtime-browser or dev-runtime-node - for example;

browser project

example: dev-runtime-browser

Verify these use cases:

1. if you will create a new repository, you can use this template, and can configure your development environment;
2. if you already have an respository cloned;

for both cases you just have run the command:

```
jspm install runtime-core=github:rethink-project/dev-runtime-core@dev-0.2
```

and on javascript code you need import the script like other modules;

```
import RuntimeUA from 'runtime-core/dist/runtimeUA';
import {Sandbox, SandboxRegistry} from 'runtime-core/dist/sandbox'
import MiniBus from 'runtime-core/dist/minibus';

console.log('Runtime: ', RuntimeUA);
console.log('Sandbox: ', Sandbox, SandboxRegistry);
console.log('MiniBus: ', MiniBus);
```

nodejs

dev-runtime-node

```
npm install github:rethink-project/dev-runtime-core#dev-0.2 --save
```

after this you can require the runtime-core like other modules on node;

```
var RuntimeUA = require('runtime-core').runtimeUA;

var runtime = new RuntimeUA();
```

if you found some issues, please submit them into the respective repository;

Karma

if you have some problems starting the karma tests, try running this commands for the following order:

1. `npm uninstall karma karma-browserify karma-mocha karma-mocha-reporter karma-chrome-launcher -g`
2. `npm install karma-cli -g`
3. `npm install`
4. `jspm update`

Note

This repository is ready to start working on development of runtime-core. The code will go to the **src** folder. The unit tests will be on **test** folder, following the name standard <component>.spec.js

To run karma tests is mandatory to run **live-server** because of the mock-up's dependencies:

```
live-server --port=4000
```

Tasks

- Documentation
- Dist
- Build
- Encode

Documentation

Generate all documentation associated to runtime core;

- if you run **gulp doc** the documentation based on jsdoc3 will be generated on folder docs/jsdoc and you can interact;

  ```
  gulp doc
  ```

- if you run **gulp api** the documentation is generate based on docs/jsdoc/*.html files, and converted to markdown;

  ```
  gulp api
  ```

- if you run **gulp docx** should be generated an .docx file, but **this process should be optimized**, is not working very well;

  ```
  gulp docx
  ```

Dist

To distribute the runtime-core, you can make a distribution file.

Run the command:

```
// compact true | false;
gulp dist --compact=false
```

Build

To distribute the runtime-core, but with the source code maps, and to detect where is some error.

Run the command:

```
gulp build
```

Encode

In this repository, we have some tasks which can help you. If you need change some resource file, like an Hyperty or ProtoStub, and load it to the Hyperties.json or ProtoStubs.json, run the following command, and answer to the questions;

```
gulp compile --file=path/to/file;
```

Example

This repository have a folder with an working example of Hyperty Connector and we can send message and make a WebRTC call between remote hyperties through the vertx;

To run the demo on example folder:

- this example have a dependecy from dev-msg-node-vertx and dev-registry-domain for communication between hyperties in two distinct browsers or tabs. **At this moment you need run locally dev-msg-node-vertx and dev-registry-domain**
- you need, in the root folder, run command: `npm start`
- in your browser, access to https://127.0.0.1:8080/example

Notes

It was done an version of RuntimeCatalogue for local instances, based on the RuntimeCatalogue, and is activated by default;

Generated by ESDoc(0.4.4)