Node1new

js-node - 0.10.9

version :1.0.0-SNAPSHOT

nodejs is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications.PHP can be deployed on most web servers and as a standalone interpreter, on almost every operating system and platform free of charge.

EXTERNALMODULE

mime
Mime is mainly used for mapping API.

mocha
mocha

mocha-lcov-reporter
mocha-lcov-reporter

should
should

nodeunit
Easy unit testing for node.js and the browser

xml2js
Xml2js module is used for conversion from xml to javascript object (JSON) using sax parsing mechanism.To display the full result use console.log(util.inspect(result, false, null)), which displays the whole result.

connect
Connect is a middleware framework for node, shipping with over 18 bundled middleware and a rich selection of 3rd-party middleware

Middleware:

1)Logger request logger with custom format support
2)Csrf Cross-site request forgery protection
3)Compress Gzip compression middleware
4)BasicAuth basic http authentication
5)BodyParser extensible request body parser
6)Json application/json parser
7)Urlencoded application/x-www-form-urlencoded parser
8)Multipart multipart/form-data parser
9)CookieParser cookie parser
10)Session session management support with bundled MemoryStore
11)SessionSession cookie-based session support
12)MethodOverride faux HTTP method support
13)ResponseTime calculates response-time and exposes via X-Response-Time
14)StaticCache memory cache layer for the static() middleware
15)Static streaming static file server supporting Range and more

16)Directory directory listing middleware
17)Vhost virtual host sub-domain mapping middleware
18)Favicon efficient favicon server (with default icon)
19)Limit limit the bytesize of request bodies
20)Query automatic querystring parser, populating req.query
21)ErrorHandler flexible error handler

Internals:

1)Server prototype
2)Connect utilities
3)Node monkey patches

Express

Express is used for high performance and for high class web development for Node.js. Creating A Server: To create an instance of the express.HTTPServer, simply invoke the createServer() method. With our instance app we can then define routes based on the HTTP verbs, in this example app.get(). Creating An HTTPS Server: To initialize a express.HTTPSServer we do the same as above, however we pass an options object, accepting key, cert and the others mentioned in node's https documentation. Configuration: Express supports arbitrary environments, such as production and development. Developers can use the configure() method to setup needs required by the current environment. When configure() is called without an environment name it will be run in every environment prior to the environment specific callback.In the example below we only dumpExceptions, and respond with exception stack traces in development mode, however for both environments we utilize methodOverride and bodyParser. Note the use of app.router, which can (optionally) be used to mount the application routes, otherwise the first call to app.get(), app.post(), etc will mount the routes. Settings: Express supports the following settings out of the box: 1)Basepath Application base path used for res.redirect() and transparently handling mounted apps. 2)Views Root views directory defaulting to CWD/views 3)View engine Default view engine name for views rendered without extensions 4)View options An object specifying global view options 5)View cache Enable view caching (enabled in production) 6)Case sensitive routes Enable case-sensitive routing 7)Strict routing When enabled trailing slashes are no longer ignored 8)Jsonp callback Enable res.send() / res.json() transparent jsonp support Routing: Express utilizes the HTTP verbs to provide a meaningful, expressive routing API. For example we may want to render a user's account for the path /user/12, this can be done by defining the route below. The values associated to the named placeholders are available as req.params. Passing Route Control: We may pass control to the next matching route, by calling the third argument, the next() function. When a match cannot be made, control is passed back to Connect, and middleware continue to be invoked in the order that they are added via use(). The same is true for several routes which have the same path defined,

they will simply be executed in order until one does not call next() and decides to respond.

Express

Express is used for high performance and for high class web development for Node.js. Creating A Server: To create an instance of the express.HTTPServer, simply invoke the createServer() method. With our instance app we can then define routes based on the HTTP verbs, in this example app.get(). Creating An HTTPS Server: To initialize a express.HTTPSServer we do the same as above, however we pass an options object, accepting key, cert and the others mentioned in node's https documentation. Configuration: Express supports arbitrary environments, such as production and development. Developers can use the configure() method to setup needs required by the current environment. When configure() is called without an environment name it will be run in every environment prior to the environment specific callback.In the example below we only dumpExceptions, and respond with exception stack traces in development mode, however for both environments we utilize methodOverride and bodyParser. Note the use of app.router, which can (optionally) be used to mount the application routes, otherwise the first call to app.get(), app.post(), etc will mount the routes. Settings: Express supports the following settings out of the box: 1)Basepath Application base path used for res.redirect() and transparently handling mounted apps. 2)Views Root views directory defaulting to CWD/views 3)View engine Default view engine name for views rendered without extensions 4)View options An object specifying global view options 5)View cache Enable view caching (enabled in production) 6)Case sensitive routes Enable case-sensitive routing 7)Strict routing When enabled trailing slashes are no longer ignored 8)Jsonp callback Enable res.send() / res.json() transparent jsonp support Routing: Express utilizes the HTTP verbs to provide a meaningful, expressive routing API. For example we may want to render a user's account for the path /user/12, this can be done by defining the route below. The values associated to the named placeholders are available as req.params. Passing Route Control: We may pass control to the next matching route, by calling the third argument, the next() function. When a match cannot be made, control is passed back to Connect, and middleware continue to be invoked in the order that they are added via use(). The same is true for several routes which have the same path defined, they will simply be executed in order until one does not call next() and decides to respond.

qs

Query string parser for node supporting nesting.This module provides utilities for dealing with query strings.Query string parser for node supporting nesting.This module provides utilities for dealing with query strings.

Methods:
1)querystring.stringify(obj, sep='&', eq='=').
2)Serialize an object to a query string. Optionally override the default separator and assignment characters.

3)querystring.parse(str, sep='&', eq='=').
4)Deserialize a query string to an object. Optionally override the default separator and assignment characters. querystring.escape.
5)The escape function used by querystring.stringify, provided so that it could be overridden if necessary. querystring.unescape.
6)The unescape function used by querystring.parse, provided so that it could be overridden if necessary.

connect

Connect is a middleware framework for node, shipping with over 18 bundled middleware and a rich selection of 3rd-party middleware

Middleware:

1)Logger request logger with custom format support
2)Csrf Cross-site request forgery protection
3)Compress Gzip compression middleware
4)BasicAuth basic http authentication
5)BodyParser extensible request body parser
6)Json application/json parser
7)Urlencoded application/x-www-form-urlencoded parser
8)Multipart multipart/form-data parser
9)CookieParser cookie parser
10)Session session management support with bundled MemoryStore
11)SessionSession cookie-based session support
12)MethodOverride faux HTTP method support
13)ResponseTime calculates response-time and exposes via X-Response-Time
14)StaticCache memory cache layer for the static() middleware
15)Static streaming static file server supporting Range and more
16)Directory directory listing middleware
17)Vhost virtual host sub-domain mapping middleware
18)Favicon efficient favicon server (with default icon)
19)Limit limit the bytesize of request bodies
20)Query automatic querystring parser, populating req.query
21)ErrorHandler flexible error handler

Internals:

1)Server prototype
2)Connect utilities
3)Node monkey patches

Express

Express is used for high performance and for high class web development for

Node.js. Creating A Server: To create an instance of the express.HTTPServer, simply invoke the createServer() method. With our instance app we can then define routes based on the HTTP verbs, in this example app.get(). Creating An HTTPS Server: To initialize a express.HTTPSServer we do the same as above, however we pass an options object, accepting key, cert and the others mentioned in node's https documentation. Configuration: Express supports arbitrary environments, such as production and development. Developers can use the configure() method to setup needs required by the current environment. When configure() is called without an environment name it will be run in every environment prior to the environment specific callback.In the example below we only dumpExceptions, and respond with exception stack traces in development mode, however for both environments we utilize methodOverride and bodyParser. Note the use of app.router, which can (optionally) be used to mount the application routes, otherwise the first call to app.get(), app.post(), etc will mount the routes. Settings: Express supports the following settings out of the box: 1)Basepath Application base path used for res.redirect() and transparently handling mounted apps. 2)Views Root views directory defaulting to CWD/views 3)View engine Default view engine name for views rendered without extensions 4)View options An object specifying global view options 5)View cache Enable view caching (enabled in production) 6)Case sensitive routes Enable case-sensitive routing 7)Strict routing When enabled trailing slashes are no longer ignored 8)Jsonp callback Enable res.send() / res.json() transparent jsonp support Routing: Express utilizes the HTTP verbs to provide a meaningful, expressive routing API. For example we may want to render a user's account for the path /user/12, this can be done by defining the route below. The values associated to the named placeholders are available as req.params. Passing Route Control: We may pass control to the next matching route, by calling the third argument, the next() function. When a match cannot be made, control is passed back to Connect, and middleware continue to be invoked in the order that they are added via use(). The same is true for several routes which have the same path defined, they will simply be executed in order until one does not call next() and decides to respond.

formidable

Formidable module is used for file uploads and form data .A node.js module for parsing form data, especially file uploads

Features:
1)Fast (~500mb/sec), non-buffering multipart parser.
2)Automatically writing file uploads to disk.
3) Low memory footprint.
4)Graceful error handling.
5) Very high test coverage.

hashish

Hash data structure manipulation functions.Hashish is a node.js library for manipulating hash data structures. It is distilled from the finest that ruby, perl, and haskell have to offer by way of hash/map interfaces.Hashish provides a chaining interface.

Methods:
1)forEach( )
2)map( )
3)filter( )
4)detect( )
5)reduce( )
6)some( )
7)updateAll( )
8)valuesAt( )
9)tap( )

hashring

Hashring provides consistent hashing based on the libketema library. The constructor is designed to handle multiple arguments types as the hash ring can be used for different use cases. The ability to use a "String" to add a single server, a "Array" to provide multiple servers or an "Object" to provide servers with a custom weight. The weight can be used to give a server a bigger distribution in the hash ring. For example you have 3 machines, 2 of those machines have 8 gig memory and one has 32 gig of memory because the last server has more memory you might it to handle more keys than the other server. So you can give it a weight of 2 and the other servers a weight of 1.

lingo

Lingo is a linguistics module, currently providing inflection and string transformations. Lingo has an ability to extends its capabilities and add additional languages.Lingo is an open source indexing system for research and teachings.

Functions of Lingo:
1)Identification of (i.e. reduction to) basic word form by means of dictionaries and suffix lists.
2)Algorithmic decomposition.
3)Dictionary-based synonymisation and identification of phrases.
4)Generic identification of phrases/word sequences based on patterns of word classes.

Lines with an asterisk (*) and lines without. That's because Lingo distinguishes between commands and data. The +text_reader+ did not only read the content of the file, but also communicated through the commands when a file begins and when it ends. This can (and will) be an important piece of information for other attendees.

mime
Mime is mainly used for mapping API.

moment
Moment is used for date formating. A lightweight javascript date library for parsing, manipulating.Moment was designed to work in all browser and in NodeJS. Instead of modifying the native Date.prototype, Moment.js creates a wrapper for the Date object. This is a pretty robust function for adding time to an existing date.

Functions:
1)An integer value representing the number of milliseconds since 1 January 1970 00:00:00 UTC.
2)A string that can be parsed by Date.parse.
3)To get the current date and time, just call moment()with no parameters.
4)An array mirroring the parameters passed into new Date().
5)Any value past the year is optional, and will default to the lowest possible number.
6)Construction with an array will create a date in the current timezone.

mysql
Mysql is mainly used for database creation .A pure node.js JavaScript Client implementing the MySQL protocol.

Table manipulation:
1)Create.
2)Update/Edit.
3)Delete.
4)Truncate.
5)Drop.

qs
Query string parser for node supporting nesting.This module provides utilities for

dealing with query strings.Query string parser for node supporting nesting.This module provides utilities for dealing with query strings.

Methods:
1)querystring.stringify(obj, sep='&', eq='=').
2)Serialize an object to a query string. Optionally override the default separator and assignment characters.
3)querystring.parse(str, sep='&', eq='=').
4)Deserialize a query string to an object. Optionally override the default separator and assignment characters. querystring.escape.
5)The escape function used by querystring.stringify, provided so that it could be overridden if necessary. querystring.unescape.
6)The unescape function used by querystring.parse, provided so that it could be overridden if necessary.

Sequelize

Sequelize library provides easy access to a MySQL database by mapping database entries to objects and vice versa. it's an ORM (Object-Relational-Mapper). The library is written entirely in JavaScript and can be used in the Node.JS environment.

Features:
1)Schema definition.
2)Schema synchronization/dropping.
3)Easy definition of class/instance methods.
4)Instance saving/updating/dropping.
5)Asynchronous library.
6)Associations.
7)Importing definitions from single files.

traverse

Traverse and transform objects by visiting every node on a recursive walk.

Methods:
1).map(fn)
2).forEach(fn)
3).reduce(fn, acc)
4).paths()
5).nodes()
6).clone()
7).get(path)
8).set(path, value)

9).has(path)

underscore

Underscore is a utility-belt library for JavaScript that provides support for the usual functional suspects (each, map, reduce, filter) without extending any core JavaScript objects. This module is used for utility purpose. Underscore provides 60-odd functions that support both the usual functional suspects [map, select, invoke — as well as more specialized helpers: function binding, javascript templating, deep equality testing,] It delegates to built-in functions and it present some of the modern browsers will use the native implementations for each, map, reduce, filter, every, some and indexOf.

underscore_string

String manipulation extensions for Underscore.js javascript library.Underscore.string is JavaScript library for comfortable manipulation with strings, extension for Underscore.js inspired by Prototype.js, Right.js, Underscore and beautiful Ruby language.

Underscore.string provides you several useful functions:
1)capitalize.
2)clean.
3)includes.
4)count.
5)escapeHTML.
6)unescapeHTML.
7)insert.
8)splice.
9)startsWith.
10)endsWith.
11) titleize.
12)trim.
13)truncate and so on.

validator

Validator is used for String validation and sanitization in JavaScript.Often it's more desirable to check or automatically sanitize parameters by name (rather than the actual string). See this gist for instructions on binding the library to the request prototype. If you are using the express.js framework you can use the express-validator middleware to seamlessly integrate node-validator.

Functions:
1)When adding to the Validator prototype, use this.str to access the string and this.error(this.msg || default_msg) when the string is invalid.
2)When adding to the Filter (sanitize) prototype, use this.str to access the string and this.modify(new_str) to update it.
3)By default, the validation methods throw an exception when a check fails.
4)To set a custom error message, set the second param of.
5)To attach a custom error handler, set the error method of the validator instance.