github-api

Version 6.2

eu90h

August 11, 2015

`(require github-api)` package: `github-api`

github-api is a wrapper for easily making requests to the GitHub api.

While this document contains usage examples, the functional tests found in the functional_tests directory of the github repository provide a good source of usage examples and patterns.

1 Authentication & Initialization

Before you begin making requests to the GitHub api, you must create an identity.

```
(struct github-identity (type data)
    #:extra-constructor-name make-github-identity)
  type : symbol?
  data : list?
```

This struct holds your GitHub identity.

`type` must be one of the following symbols: `'password` `'personal-access-token` `'oauth`

`'password` authentication simply uses your GitHub username and password.

`'personal-access-token` authentication allows you to send your GitHub username and a personal access token (created on your GitHub settings page.)

`'oauth` uses an OAuth token for authorization.

For more information, see the github api documentation

The `data` field is a list whose contents are determined by user authentication method.

For `'password` or `'personal-token` types, your data will be of the form `(list username password-or-token)`, where both `username` & `password-or-token` are strings.

For `'oauth`, the data will simply be `(list oauth-token)`, where `oauth-token` is a string.

```
(github-api  id
             [#:endpoint endpoint
              #:user-agent user-agent]) → github-api-req/c
  id : github-identity?
  endpoint : string? = "api.github.com"
  user-agent : string? = "racket-github-api-@eu90h"
```

Once you've created an identity, apply it to this procedure to receive a function for making api requests.

The optional `#:endpoint` keyword sets the root endpoint for making api requests. If you have a GitHub enterprise account, you may wish to change the endpoint. See this for more information on root-endpoints.

If you change the user-agent string, be aware that GitHub has certain rules explicated here

```
(struct github-response (code data))
```

This is a contract for the result of executing a GitHub api request. The `code` field holds the HTTP status code (in the form of a number) and `data` holds the response message in the form of a jsexpr.

```
(github-api-req/c (-> string?
                      [#:method string?
                       #:data string?
                       #:media-type string?]
                      github-response?))
```

This is a contract for the procedures returned by the function `github-api`. These functions are called with an api request and return a JSON object or a HTTP status code string. Typically, one would not use this procedure directly but rather pass it along to another function.

The `#:method` keyword specifies what HTTP verb to use (I.e. "GET", "POST", "PATCH", etc.)

The `#:data` keyword specifies any information to send along with the request. This is almost always a JSON string.

Finally, `#:media-type` specifies the format in which you wish to receive data. Practically every `github-*` procedure has an optional keyword `#:media-type` that allows you to specify a media-type for a request.

For more information on media types see the GitHub api documentation.

2 A Note on Identity Security

According to the GitHub documentation, personal access tokens are equivalent to your password. Never give it out (and don't accidently commit your identity!)

Read more about your options for authentication here

3 Example

```
(define personal-token "fs52knf535djbfk2je43b2436")
(define username "alice")
(define id (github-identity 'personal-token (list username personal-
token)))

(define github-req (github-api id))
(github-req "/users/plt/repos")
```

Here we make a request to get the repositories created by the user plt.

4 Working with JSON Data

When making requests to the GitHub API, it is common to receive data encoded in the JSON format. This quick section introduces Racket's JSON handling facilities.

Racket provides a library for working with JSON data, aptly called `json`.

This is used by the Racket github-api library to encode data before returning it.

Essentially, JSON expressions are represented as hashes. The JSON object

```
{ "name": "billy bob"}
```

becomes the hash

```
(define jsexpr (make-hash (list (cons 'name "billy bob"))))
```

To get the value associated with the key 'name, use `hash-ref` like so:

```
(hash-ref jsexpr 'name)
```

which should return `"billy bob"`

To learn more about working with hashes see the Racket guide and the Racket reference on hash-tables.

5 Gist API

The Gist API will return up to 1 megabyte of content from a requested gist.

To see if a file was truncated, check whether or not the key `truncated` is `"true"`.

To get the full contents of the file, make a request to the url referenced by the key `raw_url`.

For more information on truncation see the GitHub documentation

For additional information on the Gist API, check here

```
(github-create-gist  api-req
                     files
                    [#:description description
                     #:public public
                     #:media-type media-type]) → api-resonse/c
  api-req : github-api-req/c
  files : (listof pair?)
  description : string? = ""
  public : boolean? = #f
  media-type : string? = "application/vnd.github.v3+json"
```

Creates a gist containing files given as a list of pairs `(filename contents)`. If the gist was created successfully, a `jsexpr?` is returned.

The optional keyword *description* provides a description of the gist. By default it is empty.

The optional keyword *public* determines whether or not the gist is public. By default this is `#f`.

```
(github-edit-gist  api-req
                   gist-id
                   files
                  [#:description description
                   #:media-type media-type]) → api-resonse/c
  api-req : github-api-req/c
  gist-id : string?
  files : (listof pair?)
  description : string? = ""
  media-type : string? = "application/vnd.github.v3+json"
```

Updates a gist. See `github-create-gist` for more explanation of the arguments.

To delete a file from a gist, for example `"file1.txt"`, add an entry to the *files* list like so: `(cons "file1.txt" 'delete)`.

```
(github-get-gist  api-req
                  gist-id
                 [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  gist-id : string?
  media-type : string? = "application/vnd.github.v3+json"
```

Gets the gist, returning a `jsexpr?` on success.

```
(github-list-gist-commits  api-req
                           gist-id
                          [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  gist-id : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-star-gist  api-req
                   gist-id
                  [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  gist-id : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-unstar-gist  api-req
                     gist-id
                    [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  gist-id : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-gist-starred?  api-req
                       gist-id
                      [#:media-type media-type]) → boolean?
  api-req : github-api-req/c
  gist-id : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-fork-gist  api-req
                   gist-id
                  [#:media-type media-type]) → github-response?
```

```
  api-req : github-api-req/c
  gist-id : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-gist-forks  api-req
                         gist-id
                        [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  gist-id : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-delete-gist  api-req
                     gist-id
                    [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  gist-id : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-gist-revision  api-req
                           gist-id
                           sha
                          [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  gist-id : string?
  sha : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-user-gists  api-req
                        user
                       [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  user : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-my-gists  api-req
                      [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-my-starred-gists  api-req
                              [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-all-public-gists  api-req
                              [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

6 Gist Examples

```
(define new-gist-id
  (let ([response (github-create-gist github-req
                                      (list (cons "file1.txt" "blah
blah blah")
                                            (cons "file2.txt" "yadda
yadda yadda")))])
    (hash-ref response 'id)))

(github-edit-gist github-req new-gist-id
    (list (cons "file2.txt" 'delete)))

(github-star-gist github-req new-gist-id)
(github-gist-starred? github-req new-gist-id)
(github-unstar-gist github-req new-gist-id)
(github-gist-starred? github-req new-gist-id)

(github-fork-gist github-req new-gist-id)
(github-list-gist-forks github-req new-gist-id)

(github-get-user-gists github-req username)
```

7 Events

For more information on the Events API, see the GitHub documentation

```
(github-list-events  api-req
                     repo-owner
                     repo
                    [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-issue-events  api-req
                           repo-owner
                           repo
                          [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-public-org-events  api-req
                                org
                               [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  org : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-user-received-events  api-req
                                   user
                                  [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  user : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-user-received-public-events
 api-req
 user
 [#:media-type media-type])
```

```
 → github-response?
  api-req : github-api-req/c
  user : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-user-events  api-req
                          user
                         [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  user : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-user-public-events  api-req
                                 user
                                [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  user : string?
  media-type : string? = "application/vnd.github.v3+json"
```

8 Feeds

For more information about feeds, go here

```
(github-list-feeds  api-req
                    [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-notifications  api-req
                            [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

9 Issues

For more information about the Issues API, click here

Furthermore, the Issues API uses custom media types. See this section

```
(github-list-all-issues  api-req
                         [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-my-issues  api-req
                        [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-org-issues  api-req
                         organization
                         [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  organization : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-issues  api-req
                     repo-owner
                     repo-name
                     [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-create-issue  api-req
                      repo-owner
                      repo-name
                      title
                      [#:body body
                       #:assignee assignee
                       #:milestone milestone
                       #:labels label
                       #:media-type media-type])
```

```
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  title : string?
  body : string? = ""
  assignee : string? = ""
  milestone : string? = ""
  label : (listof string?) = null
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-edit-issue  api-req
                    repo-owner
                    repo-name
                   [#:title title
                    #:body body
                    #:assignee assignee
                    #:milestone milestone
                    #:labels label
                    #:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  title : string? = ""
  body : string? = ""
  assignee : string? = ""
  milestone : string? = ""
  label : (listof string?) = null
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-issue  api-req
                   repo-owner
                   repo-name
                   issue-number
                  [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  issue-number : (or/c number? string?)
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-issue-comments  api-req
                             repo-owner
                             repo-name
                             issue-number
                            [#:media-type media-type])
```

```
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  issue-number : (or/c number? string?)
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-comments  api-req
                       repo-owner
                       repo-name
                      [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-comment  api-req
                     repo-owner
                     repo-name
                     comment-id
                    [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  comment-id : (or/c number? string?)
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-create-comment  api-req
                        repo-owner
                        repo-name
                        issue-number
                        comment-body
                       [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  issue-number : (or/c number? string?)
  comment-body : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-edit-comment  api-req
                      repo-owner
                      repo-name
                      comment-id
                      comment-body
                     [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  comment-id : (or/c number? string?)
  comment-body : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-delete-comment  api-req
                        repo-owner
                        repo-name
                        comment-id
                       [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  comment-id : (or/c number? string?)
  media-type : string? = "application/vnd.github.v3+json"
```

10 Issue Examples

```
(github-create-issue github-req
                     "eu90h"
                     "racket-github-api"
                     "testing-issues-api"
                     #:body "this is a test of the issues api"
                     #:assignee "eu90h"
                     #:labels (list "woo!" "test"))
```

11 Repositories

```
(github-list-assignees  api-req
                        repo-owner
                        repo-name
                       [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-check-assignee  api-req
                        repo-owner
                        repo-name
                        user
                       [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  user : string?
  media-type : string? = "application/vnd.github.v3+json"
```

12 Git Data

Click here for more information on the Git Data API.

```
(github-get-blob  api-req
                  repo-owner
                  repo-name
                  sha
                 [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  sha : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-create-blob  api-req
                     repo-owner
                     repo-name
                     content
                    [encoding
                     #:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  content : string?
  encoding : string? = "utf-8"
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-commit  api-req
                    repo-owner
                    repo-name
                    sha
                   [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo-name : string?
  sha : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-create-commit  api-req
                       repo-owner
                       repo-name
                       message
                       tree
                       parents
                      [#:media-type media-type])
```

```
→ github-response?
 api-req : github-api-req/c
 repo-owner : string?
 repo-name : string?
 message : string?
 tree : string?
 parents : (listof string?)
 media-type : string? = "application/vnd.github.v3+json"
```

13 Organizations

For more on Organizations, go "https://developer.github.com/v3/orgs/" "here"

```
(github-list-orgs  api-req
                   [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-all-orgs  api-req
                       [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-user-orgs  api-req
                        user
                        [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  user : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-org  api-req
                 org
                 [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  org : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-org-members  api-req
                          org
                          [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  org : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-list-pull-requests  api-req
                            repo-owner
                            repo
                            [#:media-type media-type])
 → github-response?
```

```
api-req : github-api-req/c
repo-owner : string?
repo : string?
media-type : string? = "application/vnd.github.v3+json"
```

14 Users

```
(github-get-user  api-req
                  user
                 [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  user : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-authenticated-user  api-req
                                [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-all-users  api-req
                       [#:media-type media-type])
 → github-response?
  api-req : github-api-req/c
  media-type : string? = "application/vnd.github.v3+json"
```

15 Webhooks & Service Hooks

Webhooks are a sort-of user defined callback in the form of a listening webserver that github sends a message to whenever a certain type of event occurs.

A service hook is a webhook whose type is anything except `"web"`

To read more, see the GitHub documentation

```
(github-build-webhook-config  api-req
                              url
                             [#:content-type content-type
                              #:secret secret
                              #:insecure-ssl insecure-ssl])
 → github-response?
  api-req : github-api-req/c
  url : string?
  content-type : string? = "form"
  secret : string? = ""
  insecure-ssl : string? = "0"
```

```
(github-hook-repo  api-req
                   repo-owner
                   repo
                   type
                   config
                  [#:events events
                   #:active active]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo : string?
  type : string?
  config : jsexpr?
  events : (listof string?) = '("push")
  active : boolean? = #t
```

The `type` parameter must be the string `"web"` or a service name defined in this rather inconvenient JSON file.

Passing any other string results in an error response from the GitHub API.

Note: The `type` parameter is referred to in the GitHub documentation (misleadingly, I think) as the name of the webhook.

```
(github-get-hooks  api-req
                   repo-owner
                   repo
                  [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo : string?
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-get-hook  api-req
                  repo-owner
                  repo
                  hook-id
                 [#:media-type media-type]) → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo : string?
  hook-id : (or/c string? number?)
  media-type : string? = "application/vnd.github.v3+json"
```

```
(github-test-push-hook api-req
                       repo-owner
                       repo
                       hook-id)   → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo : string?
  hook-id : (or/c string? number?)
```

```
(github-ping-hook api-req
                  repo-owner
                  repo
                  hook-id)   → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo : string?
  hook-id : (or/c string? number?)
```

```
(github-delete-hook api-req
                    repo-owner
                    repo
                    hook-id)   → github-response?
  api-req : github-api-req/c
  repo-owner : string?
  repo : string?
  hook-id : (or/c string? number?)
```

16 Webhooks Example

```
(define config (github-build-webhook-config "http://example.com"
                                            #:insecure-ssl "0"
                                            #:content-
type "json"))
(define hook-data (github-hook-repo github-req username my-
repo "web" config))
(define hook-id (hash-ref hook-data 'id))
(github-get-hooks github-req username my-repo)
(define del-hook (thunk (github-delete-hook github-
req username my-repo hook-id)))
...
(define delete-response (del-hook))
(if (and (string? delete-response) (= 204 (get-status-code delete-
response)))
    (displayln "successfully removed webhook")
    (displayln "trouble removing webhook"))
```