Basic Browser Usage

new_browser()!
Creates new browser. This command must be called before any other, which uses browser.

browser()!

Creates new browser. Does nothing, if browser already exists.

load(url)!

Loads page in current browser. Url is passed as string. Function is finished only after page is load completely and document.readyState equals "complete".

Example. Loads google page

```
load("https://google.com")!
```

screenshot(path)!

Saves entire page bitmap file.

url()!

Returns current url as string

Example. Prints current url:

```
url()!
log(_result())
```

Example. Finish thread if url contains google.com:

```
url()!
if(_result().indexOf("google")>=0)
    success()
```

get_cookies(url)!

Returns cookies string for given url.

page()

Returns current page object.

Example. Prints google.com page content:

```
load("http://google.com")!
page().xml()!
log(_result())
```

script(script_text)!

Executes custom javascript on page. jquery can be accessed with "jQuery" variable. jquery is present even on sites, which doesn't have one

Example. Show alert box:

```
script("alert(999)")!
```

script function tries to parse return value and convert it to string, so return value should not be jquery object

Example. Hide all references:

```
script("jQuery('a').hide();0;")!
```

agent(user_agent_text)!

Sets user agent

Example. Set user agent chrome v32:

```
agent("Mozilla/5.0 (Windows NT 6.2; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/32.0.1667.0 Safari/537.36")!
```

Example. Fetches agent from resource named "agent":

```
R("agent")!
agent(_result().get())!
```

header(header_name, header_value)!

Sets arbitrary request header

Example. Set accept language to english:

```
header("Accept-Language","en-US")!
```

clear_header()!

Resets all headers to default. The only exception is user agent(see agent)

Browser Network

proxy(proxy_string)!
Parse string and set proxy. Line may have plenty of different formats parser is quite smart to recognize them all.

http and socks5 proxy are supported. socks4 is not supported.

Example. Set http proxy by ip and port

```
proxy("123.123.123.123:8080")!
```

Example. Set http proxy by host and port

```
proxy("localhost:8080")!
```

Example. Set socks5 proxy

```
proxy("socks://localhost:8080")!
```

Example. Set socks5 proxy

```
proxy("localhost:8080:socks5")!
```

Example. Set proxy with username and pass

```
proxy("localhost:8080:name:pass:socks5")!
```

Example. Set proxy with username and pass

```
proxy("http://name@pass:localhost:8080")!
```

set_proxy(server, port, is_http, name, password)!

Sets proxy. Params:

server - string, which contains server name or ip

port - integer, which contains port

is_http - boolean, if true - proxy has http type, if false - it has socks type.

name - string with username, in cause, if proxy requires authentication, otherwise blank

password - string with password, in cause, if proxy requires authentication, otherwise blank

cache_allow(wildcard)!

Used to allow caching for urls, which matches specified wildard.

If you cache specific url, you can access it later.

Caching can be used to capture image like captcha or to capture other data, like json, xml etc.

cache_allow function can be used several times to allow several url patterns.

By default no caching is allowed.

Function should be called before load(url)!.

Example. Solve captcha on yandex

```
new_browser()!
cache_allow("*.captcha.yandex.net*")!
load("https://passport.yandex.ru/registration/mail")!
solve("manual", "*.captcha.yandex.net*")!
```

cache_deny(wildcard)!

Used to deny caching for urls, which matches specified wildard.

See cache_allow for more details about caching.

Function should be called before load(url)!.

Example. Cache all data except png and jpg images

```
cache_allow("*")!
cache_deny("*.png")!
cache_deny("*.jpg")!
```

cache_get_base64(wildcard)!

Gets base64 string of cached resource.

Example. Capture and print base64 of xml, which is loaded by browser

```
    new_browser()!
    cache_allow("*.xml")!
    load("url which refers xml")!
    cache_get_base64("*.xml")!
    log(_result())
```

cache_get_string(wildcard)!

Gets string representation of cached resource.

Example. Capture and parse json response and show "valid" property of returned json

```
    new_browser()!
    cache_allow("*test.json")!
    load("url which downloads test.json via ajax")!
    cache_get_string("*test.json")!
    log(JSON.parse(_result())["valid"])
```

request_allow(wildcard)!

Allow requests to urls, which matches specified wildard.

Function should be called before load(url)!.

Example. Tell browser to load only php pages

```
request_deny("*")!
request_allow("*.php*")!
```

request_deny(wildcard)!

Denies requests to urls, which matches specified wildard.

By default all urls are loaded.

Function should be called before load(url)!.

Example. Tell browser not to load images

```
request_deny("*.png")!
request_deny("*.jpeg")!
request_deny("*.jpg")!
request_deny("*.gif")!
```

cache_data_clear()!

Clear all captured data.

This function should be called after cache is captured and processed.

cache_masks_clear()!

Clear all rules, which is set by functions cache_allow(wildcard)!, cache_deny(wildcard)!, request_allow(wildcard)!, request_deny(wildcard)!.

cache_clear()!

Clear all rules and data of caching. Equivalent calling

```
cache_masks_clear()!
cache_data_clear()!
```

is_load(wilcard)!

This function returns boolean value, which indicates if url matched by wilcard was loaded during request

Example. Test if captcha is present for current ip for google

```
load("https://www.google.com/search?q=test")!
is_load("*sorry/image*")!
if(_result())
    fail("captcha present")
```

Browser Selectors

css(selector)
Makes css query. Returns first found element.

Example. Clicks on first link:

```
css("a").click()!
```

Example. Prints content of element with class alert:

```
css(".alert").text()!
log(_result())
```

match(selector)

Searches for element by text. Selector is any text within element markup. Consider this example:

```
<a id="ref" class="c">
    Follow link
</a>
```

Then any of following selectors will match:

```
Follow
Follow link
id="ref"
<a id
<a id="ref" class="c">
```

all(selector)

Makes css query. Returns all matched elements.

This function is same as css with only difference - it returns array object.

This object has two methods: at(index)! and length()!.

at(index)! function returns element on given index length()! returns array length

Example. Prints every reference on page:

```
all("a").length()!
var len = _result()
for(var i = 0;i < len;i++)
{
    all("a").at(i).attr("href")!
    log(_result())
}
```

Nested queries are also acceptable

Example. Prints 6-th reference text inside element with id "el"

```
all("#el").at(5).css("a").text()!
log(_result())
```

match_all(selector)

Searches for element by text. Returns all matched elements.

This function is same as match with only difference - it returns array object.

This object has two methods: at(index)! and length()!.

at(index)! function returns element on given index

length()! returns array length

See all for example

Http Client

css(selector)
Makes css query. Returns first found element.

Example. Clicks on first link:

```
css("a").click()!
```

Example. Prints content of element with class alert:

```
css(".alert").text()!
log(_result())
```

match(selector)

Searches for element by text. Selector is any text within element markup. Consider this example:

```
<a id="ref" class="c">
    Follow link
</a>
```

Then any of following selectors will match:

```
Follow
Follow link
id="ref"
<a id
<a id="ref" class="c">
```

all(selector)

Makes css query. Returns all matched elements.

This function is same as css with only difference - it returns array object.

This object has two methods: at(index)! and length()!.

at(index)! function returns element on given index

length()! returns array length

Example. Prints every reference on page:

```
all("a").length()!
var len = _result()
for(var i = 0;i < len;i++)
{
    all("a").at(i).attr("href")!
    log(_result())
}
```

Nested queries are also acceptable

Example. Prints 6-th reference text inside element with id "el"

```
all("#el").at(5).css("a").text()!
log(_result())
```

match_all(selector)

Searches for element by text. Returns all matched elements.

This function is same as match with only difference - it returns array object.

This object has two methods: at(index)! and length()!.

at(index)! function returns element on given index

length()! returns array length

See all for example

Web Elements

xml()!
Returns element xml as string.

Example. Prints whole page content:

```
page().xml()!
log(_result())
```

Example. Prints content of div with id "message":

```
css("#message").xml()!
log(_result())
```

Output will be something like this:

```
<div id="message">Login successful</div>
```

If you want to get content without markup use text()! function instead.

text()!

Returns element content.

Example. Prints content of div with id "message":

```
css("#message").text()!
log(_result())
```

Output will be something like this:

```
Login successful
```

script(javascript_text)!

Executes javascript, returns result as string.

"this" points to object which calls "script" function inside javascript context

"script" function tries to parse return value and convert it to string, so return value should not be jquery object

Example. Removes element with id equals "message":

```
css("#message").script("jQuery(this).remove();0;")!
```

Example. Prints value of input element with id "pass":

```
css("#pass").script("jQuery(this).val()")!
log(_result())
```

click()!

Simulate click event on button or reference.

Example. Click on any reference on page:

```
all("a").length()!
var len = _result()
all("a").at(rand(0,len-1)).click()!
```

clear()!

Clear target input content.

type(text)!

Fills target input element on page with text. List of allowed elements: input[text], input[password], textarea

Unlike fill function, it emulates every javscript event and make pauses between every letter input.

Example. Search in google:

```
load("http://google.com")!
css(".tiah").type("test")!
css(".lsb").click()!
```

fill(text)!

Fills target input element on page with text. List of allowed elements: input[text], input[password], textarea

"fill" function acts instantly. If you want user input emulation use type

style(property)!

Returns style of target element

Example. Finds out if element with id "failure" is visible:

```
css("#failure").style("display")!
if(_result() != "none")
        success()
```

set(text)!

Sets select element. Function argument can be value or text

Consider following markup:

```
<select id="month">
        <option value="m1">January</option>
        <option value="m2">February</option>
        <option value="m3">March</option>
</select>
```

Both code will set select element to February:

```
css("#month").set("m2")!
```

```
css("#month").set("February")!
```

set_random()!

Same as set, but chooses random option.

set_integer(index)!

Set option html element.

index is integer type.

Consider following markup:

```
<select id="month">
        <option value="m1">January</option>
        <option value="m2">February</option>
        <option value="m3">March</option>
</select>
```

Following code will set select element to February:

```
css("#month").set_integer(1)!
```

check()!

Checks radio button.

Consider following markup:

```
<input type="radio" name="browser" value="ie">Internet Explorer</input>
<input type="radio" name="browser" value="opera">Opera</input>
```

Following code will set check Opera tab:

```
css("input[value='ie']").check()!
```

submit()!

Submits form. If target element is not web form, then closest form will be submitted

Consider following markup:

```
<form>
        First name: <input type="text" name="firstname">
        Last name: <input type="text" name="lastname">
</form>
```

Following code will submit form:

```
css("form").submit()!

css("input[name='firstname']").submit()!
```

focus()!

Focus on target element.

exist()!

Checks if target element exist.

Very usefull function to inspect page content.

Example. Checks if element with id equals "failure" is present:

```
if(_result())
        fail("fail")
```

Example. Wait until element with id "success" appear on page :

```
_do(function(){
        css("#success").exist()!
        if(_result())
                _break();
        sleep(1000)!
    })!
```

wait_css(selector)! can be used for this.

css(selector)!

To provide complex queries css selectors are available on elements. See Browser Selectors topic for more details

match(selector)!

To provide complex queries css selectors are available on elements. See Browser Selectors topic for more details

all(selector)!

To provide complex queries css selectors are available on elements. See Browser Selectors topic for more details

match_all(selector)!

To provide complex queries css selectors are available on elements. See Browser Selectors topic for more details