qutebrowser extensions Documentation

Florian Bruhin

CONTENTS:

1	API modules	1
	1.1 cmdutils module	1
	1.2 apitypes module	3
	1.3 config module	
	1.4 downloads module	
	1.5 hook module	5
	1.6 interceptor module	5
	1.7 message module	6
2	Tab API	7
3	Web element API	13
4	Indices and tables	15
Ру	ython Module Index	17
In	ndex	19

CHAPTER

ONE

API MODULES

1.1 cmdutils module

qutebrowser has the concept of functions which are exposed to the user as commands.

Creating a new command is straightforward:

```
from qutebrowser.api import cmdutils

@cmdutils.register(...)
def foo():
    ...
```

The commands arguments are automatically deduced by inspecting your function.

The types of the function arguments are inferred based on their default values, e.g., an argument foo=True will be converted to a flag -f/-foo in qutebrowser's commandline.

The type can be overridden using Python's function annotations:

```
@cmdutils.register(...)
def foo(bar: int, baz=True):
    ...
```

Possible values:

- A callable (int, float, etc.): Gets called to validate/convert the value.
- A python enum type: All members of the enum are possible values.
- A typing. Union of multiple types above: Any of these types are valid values, e.g., typing. Union[str, int].

```
exception qutebrowser.api.cmdutils.CommandError
```

Raised when a command encounters an error while running.

If your command handler encounters an error and cannot continue, raise this exception with an appropriate error message:

```
raise cmdexc.CommandError("Message")
```

The message will then be shown in the qutebrowser status bar.

Note: You should only raise this exception while a command handler is run. Raising it at another point causes qutebrowser to crash due to an unhandled exception.

```
qutebrowser.api.cmdutils.check_overflow(arg:int, ctype: str) \rightarrow None Check if the given argument is in bounds for the given type.
```

Parameters

- arg The argument to check.
- **ctype** The C++/Qt type to check as a string ('int'/'int64').

```
 \begin{tabular}{ll} qutebrowser.api.cmdutils.check\_exclusive (\it{flags: Iterable[bool]}, names: Iterable[str]) \rightarrow \\ None \\ \end{tabular}
```

Check if only one flag is set with exclusive flags.

Raise a CommandError if not.

Parameters

- **flags** The flag values to check.
- names A list of names (corresponding to the flags argument).

Decorator to register a new command handler.

```
class qutebrowser.api.cmdutils.argument(argname: str, **kwargs)
    Decorator to customize an argument.
```

You can customize how an argument is handled using the @cmdutils.argument decorator after @cmdutils.register. This can, for example, be used to customize the flag an argument should get:

```
@cmdutils.register(...)
@cmdutils.argument('bar', flag='c')
def foo(bar):
    ...
```

For a str argument, you can restrict the allowed strings using choices:

```
@cmdutils.register(...)
@cmdutils.argument('bar', choices=['val1', 'val2'])
def foo(bar: str):
    ...
```

For typing. Union types, the given choices are only checked if other types (like int) don't match.

The following arguments are supported for @cmdutils.argument:

- flag: Customize the short flag (-x) the argument will get.
- value: Tell qutebrowser to fill the argument with special values:
 - value=cmdutils. Value.count: The count given by the user to the command.
 - value=cmdutils.Value.win_id: The window ID of the current window.
 - value=cmdutils.Value.cur_tab: The tab object which is currently focused.
- completion: A completion function to use when completing arguments for the given command.
- choices: The allowed string choices for the argument.

The name of an argument will always be the parameter name, with any trailing underscores stripped and underscores replaced by dashes.

```
class qutebrowser.api.cmdutils.KeyMode
  Key input modes.
```

```
normal = 1
          Normal mode (no mode was entered)
     hint = 2
          Hint mode (showing labels for links)
     command = 3
          Command mode (after pressing the colon key)
     yesno = 4
          Yes/No prompts
     prompt = 5
          Text prompts
     insert = 6
          Insert mode (passing through most keys)
     passthrough = 7
          Passthrough mode (passing through all keys)
          Caret mode (moving cursor with keys)
qutebrowser.api.cmdutils.Value
     alias of qutebrowser.utils.usertypes.CommandValue
1.2 apitypes module
A single tab.
class qutebrowser.api.apitypes.ClickTarget
     How to open a clicked link.
     normal = 0
          Open the link in the current tab
     tab = 1
          Open the link in a new foreground tab
     tab bq = 2
          Open the link in a new background tab
     window = 3
          Open the link in a new window
     hover = 4
          Only hover over the link
class qutebrowser.api.apitypes.InitContext(data_dir, config_dir, args)
     Context an extension gets in its init hook.
class qutebrowser.api.apitypes.JsWorld
     World/context to run JavaScript code in.
     main = 1
          Same world as the web page's JavaScript.
     application = 2
          Application world, used by qutebrowser internally.
     user = 3
          User world, currently not used.
```

iseval = 4

World used for the jseval-command.

```
qutebrowser.api.apitypes.Tab
    alias of qutebrowser.browser.browsertab.AbstractTab

qutebrowser.api.apitypes.WebElemError
    alias of qutebrowser.browser.webelem.Error

qutebrowser.api.apitypes.WebElement
    alias of qutebrowser.browser.webelem.AbstractWebElement

exception qutebrowser.api.apitypes.WebTabError
    Base class for various errors.
```

1.3 config module

Access to the qutebrowser configuration.

```
qutebrowser.api.config.val = None
```

Simplified access to config values using attribute access. For example, to access the content. javascript.enabledsetting, you can do:

```
if config.val.content.javascript.enabled:
    ...
```

This also supports setting configuration values:

```
config.val.content.javascript.enabled = False
```

qutebrowser.api.config.get (name: str, url: PyQt5.QtCore.QUrl = None) \rightarrow Any Get a value from the config based on a string name.

1.4 downloads module

APIs related to downloading files.

```
class gutebrowser.api.downloads.TempDownload(item:
```

browser.browser.qtnetworkdownloads.DownloadItem)

aute-

A download of some data into a file object.

```
\label{eq:continuous} \verb| qutebrowser.api.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.downloads.dow
```

Download the given URL into a file object.

The download is not saved to disk.

Returns a TempDownload object, which triggers a finished signal when the download has finished:

```
dl = downloads.download_temp(QUrl("https://www.example.com/"))
dl.finished.connect(functools.partial(on_download_finished, dl))
```

After the download has finished, its successful attribute can be checked to make sure it finished successfully. If so, its contents can be read by accessing the fileobj attribute:

```
def on_download_finished(download: downloads.TempDownload) -> None:
    if download.successful:
        print(download.fileobj.read())
        download.fileobj.close()
```

1.5 hook module

Hooks for extensions.

```
class qutebrowser.api.hook.init
```

Decorator to mark a function to run when initializing.

The decorated function gets called with a *qutebrowser.api.apitypes.InitContext* as argument.

Example:

```
@hook.init()
def init(_context):
    message.info("Extension initialized.")
```

class qutebrowser.api.hook.config_changed(option_filter: str = None)

Decorator to get notified about changed configs.

By default, the decorated function is called when any change in the config occurs:

```
@hook.config_changed()
def on_config_changed():
    ...
```

When an option name is passed, it's only called when the given option was changed:

```
@hook.config_changed('content.javascript.enabled')
def on_config_changed():
    ...
```

Alternatively, a part of an option name can be specified. In the following snippet, on_config_changed gets called when either bindings.commands or bindings.key_mappings have changed:

```
@hook.config_changed('bindings')
def on_config_changed():
    ...
```

1.6 interceptor module

APIs related to intercepting/blocking requests.

```
qutebrowser.api.interceptor.register(interceptor: Callable[[qutebrowser.extensions.interceptors.Request], None]) \rightarrow None
```

Register a request interceptor.

Whenever a request happens, the interceptor gets called with a Request object.

Example:

```
def intercept(request: interceptor.Request) -> None:
   if request.request_url.host() == 'badhost.example.com':
        request.block()
```

class qutebrowser.api.interceptor.Request (first_party_url, request_url, is_blocked=False)
 A request which can be intercepted/blocked.

```
first_party_url = None
    The URL of the page being shown.
```

1.5. hook module 5

```
request_url = None
The LIPL of the file be
```

The URL of the file being requested.

 $\textbf{block}\,(\,)\,\to None$

Block this request.

1.7 message module

Utilities to display messages above the status bar.

```
qutebrowser.api.message.error(message: str, *, stack: str = None, replace: bool = False) \rightarrow None Display an error message.
```

Parameters

- message The message to show.
- **stack** The stack trace to show (if any).
- replace Replace existing messages which are still being shown.

qutebrowser.api.message.info(message: str, *, replace: bool = False) \rightarrow None Display an info message.

Parameters

- message The message to show.
- replace Replace existing messages which are still being shown.

qutebrowser.api.message.warning (message: str, *, replace: bool = False) \rightarrow None Display a warning message.

Parameters

- message The message to show.
- replace Replace existing messages which are still being shown.

CHAPTER

TWO

TAB API

class qutebrowser.browser.browsertab.AbstractTab

An adapter for QWebView/QWebEngineView representing a single tab.

window_close_requested

Signal emitted when a website requests to close this tab.

link hovered

Signal emitted when a link is hovered (the hover text)

load_started

Signal emitted when a page started loading

load_progress

Signal emitted when a page is loading (progress percentage)

load finished

Signal emitted when a page finished loading (success as bool)

icon_changed

Signal emitted when a page's favicon changed (icon as QIcon)

title_changed

Signal emitted when a page's title changed (new title as str)

new_tab_requested

Signal emitted when a new tab should be opened (url as QUrl)

url changed

Signal emitted when a page's URL changed (url as QUrl)

contents_size_changed

Signal emitted when a tab's content size changed (new size as QSizeF)

fullscreen_requested

Signal emitted when a page requested full-screen (bool)

before_load_started

Signal emitted before load starts (URL as QUrl)

$send_event (evt: PyQt5.QtCore.QEvent) \rightarrow None$

Send the given event to the underlying widget.

The event will be sent via QApplication.postEvent. Note that a posted event must not be re-used in any way!

None Send a fake key event to this tab.

```
dump_async (callback: Callable[[str], None], *, plain: bool = False) \rightarrow None Dump the current page's html asynchronously.
```

The given callback will be called with the result when dumping is complete.

```
\begin{tabular}{ll} {\bf run\_js\_async}\ (code: & str, & callback: & Callable[[Any], & None] & = None, & *, & world: \\ & & Union[qutebrowser.utils.usertypes.JsWorld, int] & = None) & \to None \\ & & Run\ javascript\ async. & \\ \end{tabular}
```

The given callback will be called with the result when running JS is complete.

Parameters

- code The javascript code to run.
- callback The callback to call with the result, or None.
- world A world ID (int or usertypes.JsWorld member) to run the JS in the main world or in another isolated world.

class qutebrowser.browser.browsertab.AbstractAction

Attribute action of AbstractTab for Qt WebActions.

```
\textbf{exit\_fullscreen} \, (\,) \, \to None
```

Exit the fullscreen mode.

```
save\_page() \rightarrow None
```

Save the current page.

```
run\_string(name: str) \rightarrow None
```

Run a webaction based on its name.

```
show source (pygments: bool = False) \rightarrow None
```

Show the source of the current page in a new tab.

```
class qutebrowser.browser.browsertab.AbstractPrinting
```

Attribute printing of AbstractTab for printing the page.

```
check\_pdf\_support() \rightarrow None
```

Check whether writing to PDFs is supported.

If it's not supported (by the current Qt version), a WebTabError is raised.

```
check\_printer\_support() \rightarrow None
```

Check whether writing to a printer is supported.

If it's not supported (by the current Qt version), a WebTabError is raised.

```
\textbf{check\_preview\_support}\;()\;\rightarrow None
```

Check whether showing a print preview is supported.

If it's not supported (by the current Qt version), a WebTabError is raised.

```
to_pdf (filename: str) \rightarrow bool
```

Print the tab to a PDF with the given filename.

to_printer(printer: PyQt5.QtPrintSupport.QPrinter, callback: Callable[[bool], None] = None) \rightarrow None Print the tab.

Parameters

- **printer** The QPrinter to print to.
- callback Called with a boolean (True if printing succeeded, False otherwise)

8 Chapter 2. Tab API

show dialog() \rightarrow None

Print with a QPrintDialog.

class qutebrowser.browser.browsertab.AbstractSearch

Attribute search of AbstractTab for doing searches.

text

The last thing this view was searched for.

search displayed

Whether we're currently displaying search results in this view.

_flags

The flags of the last search (needs to be set by subclasses).

_widget

The underlying WebView widget.

finished

Signal emitted when a search was finished (True if the text was found, False otherwise)

cleared

Signal emitted when an existing search was cleared.

search (text: str, *, ignore_case: qutebrowser.utils.usertypes.IgnoreCase = <IgnoreCase.never: 2>, reverse: bool = False, result_cb: Callable[[bool], None] = None) \rightarrow None Find the given text on the page.

Parameters

- text The text to search for.
- ignore_case Search case-insensitively.
- reverse Reverse search direction.
- **result_cb** Called with a bool indicating whether a match was found.

$\texttt{clear}() \rightarrow None$

Clear the current search.

$prev_result(*, result_cb: Callable[[bool], None] = None) \rightarrow None$

Go to the previous result of the current search.

Parameters result_cb - Called with a bool indicating whether a match was found.

$$next_result(*, result_cb: Callable[[bool], None] = None) \rightarrow None$$

Go to the next result of the current search.

Parameters result_cb - Called with a bool indicating whether a match was found.

class qutebrowser.browser.browsertab.AbstractZoom

Attribute zoom of AbstractTab for controlling zoom.

$apply_offset: int) \rightarrow None$

Increase/Decrease the zoom level by the given offset.

Parameters offset – The offset in the zoom level list.

Returns The new zoom percentage.

 $\verb|set_factor| (factor: float, *, fuzzyval: bool = True) \rightarrow None$

Zoom to a given zoom factor.

Parameters

• factor – The zoom factor as float.

• **fuzzyval** – Whether to set the NeighborLists fuzzyval.

class gutebrowser.browser.browsertab.AbstractCaret

Attribute caret of AbstractTab for caret browsing.

selection_toggled

Signal emitted when the selection was toggled. (argument - whether the selection is now active)

follow selected done

Emitted when a follow selection action is done.

class qutebrowser.browser.browsertab.AbstractScroller

Attribute scroller of AbstractTab to manage scroll position.

perc_changed

Signal emitted when the scroll position changed (int, int)

before_jump_requested

Signal emitted before the user requested a jump. Used to set the special 'mark so the user can return.

class qutebrowser.browser.browsertab.AbstractHistory

The history attribute of a AbstractTab.

back (*count:* int = 1) \rightarrow None

Go back in the tab's history.

forward (*count: int* = 1) \rightarrow None

Go forward in the tab's history.

class gutebrowser.browser.browsertab.AbstractElements

Finding and handling of elements on the page.

find_css (selector: str, callback: Callable[[Sequence[webelem.AbstractWebElement]], None], error_cb: Callable[[Exception], None], *, only_visible: bool = False) \rightarrow None Find all HTML elements matching a given selector async.

If there's an error, the callback is called with a webelem. Error instance.

Parameters

- callback The callback to be called when the search finished.
- **error cb** The callback to be called when an error occurred.
- **selector** The CSS selector to search for.
- only_visible Only show elements which are visible on screen.

 $find_id$ (elem_id: str, callback: Callable[[Optional[webelem.AbstractWebElement]], None]) \rightarrow None Find the HTML element with the given ID async.

Parameters

- callback The callback to be called when the search finished. Called with a WebEngineElement or None.
- **elem_id** The ID to search for.

Parameters callback – The callback to be called when the search finished. Called with a WebEngineElement or None.

10 Chapter 2. Tab API

find_at_pos (pos: PyQt5.QtCore.QPoint, callback: Callable[[Optional[webelem.AbstractWebElement]], $None J) \rightarrow None$ Find the element at the given position async.

This is also called "hit test" elsewhere.

Parameters

- pos The QPoint to get the element for.
- callback The callback to be called when the search finished. Called with a WebEngineElement or None.

class qutebrowser.browser.browsertab.AbstractAudio

Handling of audio/muting for this tab.

 set_muted (muted: bool, override: bool = False) \rightarrow None

Set this tab as muted or not.

Parameters override - If set to True, muting/unmuting was done manually and overrides future automatic mute/unmute changes based on the URL.

$\textbf{is_recently_audible}\,(\,)\,\rightarrow bool$

Whether this tab has had audio playing recently.

datebiowsei exterisions bocamentation	gutebrowser	extensions	Documentation
---------------------------------------	-------------	------------	----------------------

12 Chapter 2. Tab API

THREE

WEB ELEMENT API

```
class qutebrowser.browser.webelem.AbstractWebElement(tab)
     A wrapper around QtWebKit/QtWebEngine web element.
     tab
          The tab associated with this element.
     has frame()
          Check if this element has a valid frame attached.
     geometry()
          Get the geometry for this element.
     classes()
          Get a list of classes assigned to this element.
     tag_name()
          Get the tag name of this element.
          The returned name will always be lower-case.
     outer xml()
          Get the full HTML representation of this element.
          Get the value attribute for this element, or None.
     set value(value)
          Set the element value.
     dispatch_event (event, bubbles=False, cancelable=False, composed=False)
          Dispatch an event to the element.
              Parameters
                   • bubbles – Whether this event should bubble.
                   • cancelable – Whether this event can be cancelled.
                   • composed – Whether the event will trigger listeners outside of a shadow root.
     insert_text(text)
          Insert the given text into the element.
     rect_on_view(*, elem_geometry=None, no_js=False)
          Get the geometry of the element relative to the webview.
              Parameters
```

• **elem_geometry** – The geometry of the element, or None.

• **no_js** – Fall back to the Python implementation.

is writable()

Check whether an element is writable.

is_content_editable()

Check if an element has a contenteditable attribute.

Parameters elem – The QWebElement to check.

Returns True if the element has a contenteditable attribute, False otherwise.

is_editable (strict=False)

Check whether we should switch to insert mode for this element.

Parameters strict – Whether to do stricter checking so only fields where we can get the value match, for use with the :editor command.

Returns True if we should switch to insert mode, False otherwise.

is_text_input()

Check if this element is some kind of text box.

remove_blank_target()

Remove target from link.

resolve url(baseurl)

Resolve the URL in the element's src/href attribute.

Parameters baseurl – The URL to base relative URLs on as QUrl.

Returns A QUrl with the absolute URL, or None.

is_link()

Return True if this AbstractWebElement is a link.

click (click_target, *, force_event=False)

Simulate a click on the element.

Parameters

- click_target A usertypes.ClickTarget member, what kind of click to simulate.
- **force_event** Force generating a fake mouse event.

hover()

Simulate a mouse hover over the element.

class gutebrowser.browser.webelem.Error

Base class for WebElement errors.

class qutebrowser.browser.webelem.OrphanedError

Raised when a webelement's parent has vanished.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

qutebrowser extensions Documentation	

PYTHON MODULE INDEX

q

```
qutebrowser.api.apitypes, 3
qutebrowser.api.cmdutils, 1
qutebrowser.api.config, 4
qutebrowser.api.downloads, 4
qutebrowser.api.hook, 5
qutebrowser.api.interceptor, 5
qutebrowser.api.message, 6
```

18 Python Module Index

INDEX

Symbols	attribute), 7
_flags (qutebrowser.browser.browsertab.AbstractSearch	block() (qutebrowser.api.interceptor.Request method), 6
attribute), 9	C
_widget (qutebrowser.browser.browsertab.AbstractSearch attribute), 9	caret (qutebrowser.api.cmdutils.KeyMode attribute), 3
attribute), 9	check_exclusive() (in module qutebrowser.api.cmdutils),
A	2
AbstractAction (class in qute-	<pre>check_overflow() (in module qutebrowser.api.cmdutils),</pre>
browser.browsertab), 8	1
AbstractAudio (class in qute-	check_pdf_support() (qute-
browser.browsertab), 11	browser.browser.browsertab.AbstractPrinting
AbstractCaret (class in qutebrowser.browser.browsertab),	method), 8
10	check_preview_support() (qute-
AbstractElements (class in qute-	browser.browser.browsertab.AbstractPrinting
browser.browsertab), 10	method), 8
AbstractHistory (class in qute-	check_printer_support() (qute-
browser.browser.browsertab), 10	browser.browser.browsertab.AbstractPrinting
AbstractPrinting (class in qute-	method), 8
browser.browsertab), 8	classes() (qutebrowser.browser.webelem.AbstractWebElement
AbstractScroller (class in qute-	method), 13
browser.browser.ab), 10	clear() (qutebrowser.browser.browsertab.AbstractSearch
AbstractSearch (class in qute-	method), 9 cleared (qutebrowser.browser.browsertab.AbstractSearch
browser.browser.browsertab), 9	attribute), 9
AbstractTab (class in qutebrowser.browser.browsertab), 7	click() (qutebrowser.browser.webelem.AbstractWebElement
AbstractWebElement (class in qute-	method), 14
browser.browser.webelem), 13	ClickTarget (class in qutebrowser.api.apitypes), 3
AbstractZoom (class in qutebrowser.browser.browsertab),	command (qutebrowser.api.cmdutils.KeyMode attribute),
9	command (queorowser.apr.emdums.Reywode attribute),
application (qutebrowser.api.apitypes.JsWorld attribute),	CommandError, 1
apply_offset() (qutebrowser.browser.browsertab.AbstractZo	
method), 9	contents_size_changed (qute-
	browser.browsertab.AbstractTab
argument (class in qutebrowser.api.cmdutils), 2	attribute), 7
В	D
back() (qutebrowser.browser.browsertab.AbstractHistory	D
method), 10	$dispatch_event() \ (qutebrowser.browser.webelem. AbstractWebElemen \ and \ dispatch_event()) \ (qutebrowser.browser.webelem. AbstractWebElemen \ and \ and$
before_jump_requested (qute-	method), 13
browser.browser.browsertab.AbstractScroller	download_temp() (in module qute-
attribute), 10	browser.api.downloads), 4
before_load_started (qute-	dump_async() (qutebrowser.browser.browsertab.AbstractTab
browser.browser.browsertab.AbstractTab	method), 7

E	is_content_editable() (qute-
Error (class in qutebrowser.browser.webelem), 14	browser. we belem. Abstract Web Element
error() (in module qutebrowser.api.message), 6	method), 14
exit_fullscreen() (qutebrowser.browser.browsertab.AbstractA method), 8	istroditable() (qutebrowser.browser.webelem.AbstractWebElement method), 14
	is_link() (qutebrowser.browser.webelem.AbstractWebElement method), 14
	is_recently_audible() (qute-
browser.browser.browsertab.AbstractTab	browser.browsertab.AbstractAudio method), 11
find_at_pos() (qutebrowser.browser.browsertab.AbstractEler	is text_input() (qutebrowser.browser.webelem.AbstractWebElement method), 14
find_css() (qutebrowser.browser.browsertab.AbstractElemen method), 10	is_writable() (qutebrowser.browser.webelem.AbstractWebElement method), 14
find_focused() (qutebrowser.browser.browsertab.AbstractElemethod), 10	•
find_id() (qutebrowser.browser.browsertab.AbstractElement method), 10	JsWorld (class in qutebrowser.api.apitypes.Js World attribute), 3
attribute), 9	K
first_party_url (qutebrowser.api.interceptor.Request attribute), 5	KeyMode (class in qutebrowser.api.cmdutils), 2
follow_selected_done (qute-	Link havened (autobuoyyeen kuoyyeen kuoyyeentah Abetue at Tah
browser.browser.browsertab.AbstractCaret attribute), 10	link_hovered (qutebrowser.browser.browsertab.AbstractTab attribute), 7
forward() (qutebrowser.browser.browsertab.AbstractHistory method), 10	load_finished (qutebrowser.browser.browsertab.AbstractTab attribute), 7
fullscreen_requested (qute- browser.browser.browsertab.AbstractTab	load_progress (qutebrowser.browser.browsertab.AbstractTab attribute), 7
attribute), 7	load_started (qutebrowser.browser.browsertab.AbstractTab attribute), 7
G	M
geometry() (qutebrowser.browser.webelem.AbstractWebEle	ment main (qutebrowser.api.apitypes.JsWorld attribute), 3
get() (in module qutebrowser.api.config), 4	N
Н	new_tab_requested (qute-
has_frame() (qutebrowser.browser.webelem.AbstractWebEle	browser.browser.browsertab.AbstractTab
method), 13	attribute), 7 next_result() (qutebrowser.browser.browsertab.AbstractSearch
hint (qutebrowser.api.cmdutils.KeyMode attribute), 3	method), 9
hover() (qutebrowser.apr.aprtypes.enek rarget attribute), 3 hover() (qutebrowser.browser.webelem.AbstractWebElemen	normal (qutebrowser.api.apitypes.ClickTarget attribute),
method) 14	normal (qutebrowser.api.cmdutils.KeyMode attribute), 2
I	0
icon_changed (qutebrowser.browser.browsertab.AbstractTab	Out of Trans (day in the outless and day)
, , , ,	OrphanedError (class in quiebrowser.orowser.webeiem),
info() (in module qutebrowser.api.message), 6 init (class in qutebrowser.api.hook), 5	outer_xml() (qutebrowser.browser.webelem.AbstractWebElement
Init (class in quiebrowser.api.nook), 3 InitContext (class in quiebrowser.api.apitypes), 3	method), 13
insert (autebrowser ani cmdutils KeyMode attribute) 3	P
insert_text() (qutebrowser.browser.webelem.AbstractWebElemethod), 13	ement passthrough (qutebrowser.api.cmdutils.KeyMode at- tribute), 3

20 Index

perc_changed (qutebrowser.browser.browsertab.AbstractSc attribute), 10	er shlew _source() (qutebrowser.browser.browsertab.AbstractAction method), 8
prev_result() (qutebrowser.browser.browsertab.AbstractSea method), 9	
prompt (qutebrowser.api.cmdutils.KeyMode attribute), 3	Tab (in module qutebrowser.api.apitypes), 4
Q	tab (qutebrowser.api.apitypes.ClickTarget attribute), 3 tab (qutebrowser.browser.webelem.AbstractWebElement
qutebrowser.api.apitypes (module), 3 qutebrowser.api.cmdutils (module), 1 qutebrowser.api.config (module), 4 qutebrowser.api.downloads (module), 4 qutebrowser.api.hook (module), 5 qutebrowser.api.interceptor (module), 5 qutebrowser.api.message (module), 6	attribute), 13 tab_bg (qutebrowser.api.apitypes.ClickTarget attribute), 3 tag_name() (qutebrowser.browser.webelem.AbstractWebElement method), 13 TempDownload (class in qutebrowser.api.downloads), 4 text (qutebrowser.browser.browsertab.AbstractSearch attribute), 9
R	title_changed (qutebrowser.browser.browsertab.AbstractTab attribute), 7
rect_on_view() (qutebrowser.browser.webelem.AbstractWe method), 13 register (class in qutebrowser.api.cmdutils), 2 register() (in module qutebrowser.api.interceptor), 5	httlendent(qutebrowser.browser.browsertab.AbstractPrinting method), 8 to_printer() (qutebrowser.browser.browsertab.AbstractPrinting method), 8
remove_blank_target() (qute- browser.browser.webelem.AbstractWebElement method), 14 Request (class in qutebrowser.api.interceptor), 5 request_url (qutebrowser.api.interceptor.Request tribute), 5	U url_changed (qutebrowser.browser.browsertab.AbstractTab attribute), 7 user (qutebrowser.api.apitypes.JsWorld attribute), 3
resolve_url() (qutebrowser.browser.webelem.AbstractWebE method), 14 run_js_async() (qutebrowser.browser.browsertab.AbstractT method), 8 run_string() (qutebrowser.browser.browsertab.AbstractActi method), 8	val (in module qutebrowser.api.config), 4 alvalue (in module qutebrowser.api.cmdutils), 3 value() (qutebrowser.browser.webelem.AbstractWebElement on method), 13
S	W
save_page() (qutebrowser.browser.browsertab.AbstractActi method), 8	warning() (in module qutebrowser.api.message), 6 WebElement (in module qutebrowser.api.apitypes), 4 WebElemError (in module qutebrowser.api.apitypes), 4
search() (qutebrowser.browser.browsertab.AbstractSearch method), 9	WebTabError, 4 window (qutebrowser.api.apitypes.ClickTarget attribute),
search_displayed (qute- browser.browser.browsertab.AbstractSearch attribute), 9	window_close_requested (qute- browser.browser.browsertab.AbstractTab
selection_toggled (qute- browser.browser.browsertab.AbstractCaret attribute), 10	attribute), 7
send_event() (qutebrowser.browser.browsertab.AbstractTab method), 7	
set_factor() (qutebrowser.browser.browsertab.AbstractZoormethod), 9 set_muted() (qutebrowser.browser.browsertab.AbstractAud	
method), 11 set_value() (qutebrowser.browser.webelem.AbstractWebEle	ement
method), 13	
show_dialog() (qutebrowser.browser.browsertab.AbstractPr method), 8	initing

Index 21