puppeteer package

Classes

Class	Description
Accessibility	The Accessibility class provides methods for inspecting Chromium's accessibility tree. The accessibility tree is used by assistive technology such as screen readers or switches.
Browser	A Browser is created when Puppeteer connects to a Chromium instance, either through PuppeteerNode.launch() or Puppeteer.connect().
BrowserContext	BrowserContexts provide a way to operate multiple independent browser sessions. When a browser is launched, it has a single BrowserContext used by default. The method Browser.newPage creates a page in the default browser context.
BrowserFetcher	BrowserFetcher can download and manage different versions of Chromium and Firefox.
CDPSession	The `CDPSession` instances are used to talk raw Chrome Devtools Protocol.
Connection	
ConsoleMessage	ConsoleMessage objects are dispatched by page via the 'console' event.
Coverage	The Coverage class provides methods to gathers information about parts of JavaScript and CSS that were used by the page.
CSSCoverage	
CustomError	
Dialog	Dialog instances are dispatched by the Page via the `dialog` event.
ElementHandle	ElementHandle represents an in-page DOM element.
EventEmitter	The EventEmitter class that many Puppeteer classes extend.
ExecutionContext	This class represents a context for JavaScript execution. A [Page] might have many execution contexts: - each frame has "default" execution context that is always created after frame is attached to DOM. This context is returned by the Frame.executionContext() method. - Extension's content scripts create additional execution contexts.Besides pages, execution contexts can be found in workers.
FileChooser	File choosers let you react to the page requesting for a file.
Frame	At every point of time, page exposes its current frame tree via the page.mainFrame and frame.childFrames methods.
HTTPRequest	Represents an HTTP request sent by a page.
HTTPResponse	The HTTPResponse class represents responses which are received by the Page class.
JSCoverage	
JSHandle	Represents an in-page JavaScript object. JSHandles can be created with the

	page.evaluateHandle method.
Keyboard	Keyboard provides an api for managing a virtual keyboard. The high level api is Keyboard.type(), which takes raw characters and generates proper keydown, keypress/input, and keyup events on your page.
Mouse	The Mouse class operates in main-frame CSS pixels relative to the top-left corner of the viewport.
Page	Page provides methods to interact with a single tab or extension background page in Chromium.
Puppeteer	The main Puppeteer class.IMPORTANT: if you are using Puppeteer in a Node environment, you will get an instance of PuppeteerNode when you import or require `puppeteer`. That class extends `Puppeteer`, so has all the methods documented below as well as all that are defined on PuppeteerNode.
PuppeteerNode	Extends the main Puppeteer class with Node specific behaviour for fetching and downloading browsers.If you're using Puppeteer in a Node environment, this is the class you'll get when you run `require('puppeteer')` (or the equivalent ES `import`).
SecurityDetails	The SecurityDetails class represents the security details of a response that was received over a secure connection.
Target	
TimeoutError	TimeoutError is emitted whenever certain operations are terminated due to timeout.
Touchscreen	The Touchscreen class exposes touchscreen events.
Tracing	The Tracing class exposes the tracing audit interface.
WebWorker	The WebWorker class represents a WebWorker.

Enumerations

Enumeration	Description
BrowserContextEmittedEvents	
BrowserEmittedEvents	All the events a browser instance may emit.
PageEmittedEvents	All the events that a page instance may emit.

Functions

Function	Description
clearCustomQueryHandlers()	Clears all registered handlers.
connect(options)	This method attaches Puppeteer to an existing browser instance.
customQueryHandlerNames()	
launch(options)	Launches puppeteer and launches a browser instance with given arguments and options when specified.

registerCustomQueryHandler(name, queryHandler)	Registers a custom query handler. After registration, the handler can be used everywhere where a selector is expected by prepending the selection string with `<name>/`. The name is only allowed to consist of lower- and upper case latin letters.
unregisterCustomQueryHandler(name)	

Interfaces

Interface	Description
BoundingBox	
BoxModel	
BrowserConnectOptions	Generic browser options that can be passed when launching any browser or when connecting to an existing browser instance.
BrowserFetcherOptions	
BrowserFetcherRevisionInfo	
BrowserLaunchArgumentOptions	Launcher options that only apply to Chrome.
CDPSessionOnMessageObject	
ClickOptions	
CommonEventEmitter	
ConnectionCallback	
ConnectionTransport	
ConnectOptions	
ConsoleMessageLocation	
ContinueRequestOverrides	
CoverageEntry	The CoverageEntry class represents one entry of the coverage report.
Credentials	
CSSCoverageOptions	Set of configurable options for CSS coverage.
CustomQueryHandler	Contains two functions `queryOne` and `queryAll` that can be registered as alternative querying strategies. The functions `queryOne` and `queryAll` are executed in the page context. `queryOne` should take an `Element` and a selector string as argument and return a single `Element` or `null` if no element is found. `queryAll` takes the same arguments but should instead return a `NodeListOf<Element>` or `Array<Element>` with all the elements that match the given query selector.
Device	
FrameAddScriptTagOptions	

FrameAddStyleTagOptions	
FrameWaitForFunctionOptions	
GeolocationOptions	
InternalNetworkConditions	
JSCoverageOptions	Set of configurable options for JS coverage.
JSONObject	
LaunchOptions	Generic launch options that can be passed when launching any browser.
MediaFeature	
Metrics	
MouseOptions	
MouseWheelOptions	
NetworkConditions	
PageEventObject	Denotes the objects received by callback functions for page events.See PageEmittedEvents for more detail on the events and when they are emitted.
PDFMargin	
PDFOptions	Valid options to configure PDF generation via Page.pdf().
Point	
PressOptions	
ProductLauncher	Describes a launcher - a class that is able to create and launch a browser instance.
PuppeteerEventListener	
RemoteAddress	
ResponseForRequest	Required response data to fulfill a request with.
ScreenshotClip	
ScreenshotOptions	
SerializedAXNode	Represents a Node and the properties of it that are relevant to Accessibility.
SnapshotOptions	
TracingOptions	
Viewport	Sets the viewport of the page.
WaitForOptions	

WaitForSelectorOptions	
WaitForTargetOptions	
WaitTimeoutOptions	

Variables

Variable	Description
devices	
errors	
EVALUATION_SCRIPT_URL	
networkConditions	
puppeteerErrors	

Type Aliases

Type Alias	Description
ActionResult	
ChromeReleaseChannel	
ConsoleMessageType	The supported types for console messages.
DevicesMap	
ErrorCode	
EvaluateFn	
EvaluateFnReturnType	
EvaluateHandleFn	
EventType	
Handler	
InterceptResolutionStrategy	
JSONArray	
KeyInput	All the valid keys that can be passed to functions that take user input, such as keyboard.press
MouseButton	
PaperFormat	All the valid paper format types when printing a PDF.
Permission	
Platform	Supported platforms.

PredefinedNetworkConditions	
Product	Supported products.
ProtocolLifeCycleEvent	
PuppeteerErrors	
PuppeteerLifeCycleEvent	
PuppeteerNodeLaunchOptions	Utility type exposed to enable users to define options that can be passed to `puppeteer.launch` without having to list the set of all types.
ResourceType	Resource types for HTTPRequests as perceived by the rendering engine.
Serializable	
SerializableOrJSHandle	
TargetFilterCallback	
UnwrapElementHandle	Unwraps a DOM element out of an ElementHandle instance
UnwrapPromiseLike	
WrapElementHandle	Wraps a DOM element into an ElementHandle instance