Page class

Page provides methods to interact with a single tab or extension background page in Chromium.

Signature:

```
export declare class Page extends EventEmitter
```

Extends: EventEmitter

Remarks

One Browser instance might have multiple Page instances.

The constructor for this class is marked as internal. Third-party code should not call the constructor directly or create subclasses that extend the `Page` class.

Example 1

This example creates a page, navigates it to a URL, and then * saves a screenshot:

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({path: 'screenshot.png'});
  await browser.close();
})();
```

The Page class extends from Puppeteer's EventEmitter class and will emit various events which are documented in the PageEmittedEvents enum.

Example 2

This example logs a message for a single page `load` event:

```
page.once('load', () => console.log('Page loaded!'));
```

To unsubscribe from events use the `off` method:

```
function logRequest(interceptedRequest) {
  console.log('A request was made:', interceptedRequest.url());
}
page.on('request', logRequest);
// Sometime later...
page.off('request', logRequest);
```

Properties

Property	Modifiers	Type	Description
accessibility		Accessibility	
coverage		Coverage	
keyboard		Keyboard	
mouse		Mouse	
touchscreen		Touchscreen	
tracing		Tracing	

Methods

Method	Modifiers	Description
$(selector)		Runs `document.querySelector` within the page. I element matches the selector, the return value resolves to `null`.
$$(selector)		The method runs `document.querySelectorAll` within the page. If no elements match the selec the return value resolves to `[]`.
$$eval(selector, pageFunction, args)		This method runs `Array.from(document.querySelectorAll(select` within the page and passes the result as the firs argument to the `pageFunction`.
$eval(selector, pageFunction, args)		This method runs `document.querySelector` withi the page and passes the result as the first argu to the `pageFunction`.
$x(expression)		The method evaluates the XPath expression rela to the page document as its context node. If the are no such elements, the method resolves to a empty array.
addScriptTag(options)		Adds a `<script>` tag into the page with the desi URL or content.
addStyleTag(options)		Adds a `<link rel="stylesheet">` tag into the pa with the desired URL or a `<style type="text/cs` tag with the content.
authenticate(credentials)		Provide credentials for `HTTP authentication`.
bringToFront()		Brings page to front (activates tab).
browser()		Get the browser the page belongs to.

browserContext()		Get the browser context that the page belongs
click(selector, options)		This method fetches an element with `selector`, scrolls it into view if needed, and then uses Page.mouse to click in the center of the elemen there's no element matching `selector`, the meth throws an error.
close(options)		
content()		
cookies(urls)		If no URLs are specified, this method returns cookies for the current page URL. If URLs are specified, only cookies for those URLs are returr
deleteCookie(cookies)		
emulate(options)		Emulates given device metrics and user agent. T method is a shortcut for calling two methods: Page.setUserAgent() and Page.setViewport() To emulation, Puppeteer provides a list of device descriptors that can be obtained via the Puppeteer.devices `page.emulate` will resize the page. A lot of websites don't expect phones to change size, so you should emulate before navigating to the page.
emulateIdleState(overrides)		Emulates the idle state. If no arguments set, clea idle state emulation.
emulateMediaFeatures(features)		
emulateMediaType(type)		
emulateNetworkConditions(networkConditions)		
emulateTimezone(timezoneId)		
emulateVisionDeficiency(type)		Simulates the given vision deficiency on the pag
evaluate(pageFunction, args)		
evaluateHandle(pageFunction, args)		
evaluateOnNewDocument(pageFunction, args)		Adds a function which would be invoked in one the following scenarios:- whenever the page is navigated- whenever the child frame is attached navigated. In this case, the function is invoked i the context of the newly attached frame.The function is invoked after the document was crea but before any of its scripts were run. This is use to amend the JavaScript environment, e.g. to se `Math.random`.
exposeFunction(name, puppeteerFunction)		The method adds a function called `name` on the

		page's `window` object. When called, the function executes `puppeteerFunction` in node.js and retu a `Promise` which resolves to the return value of `puppeteerFunction`.If the puppeteerFunction ret a `Promise`, it will be awaited.NOTE: Functions installed via `page.exposeFunction` survive navigations.
focus(selector)		This method fetches an element with `selector` a focuses it. If there's no element matching `selec` the method throws an error.
frames()		
goBack(options)		This method navigate to the previous page in history.
goForward(options)		This method navigate to the next page in histor
goto(url, options)		
hover(selector)		This method fetches an element with `selector`, scrolls it into view if needed, and then uses Page.mouse to hover over the center of the element. If there's no element matching `select` the method throws an error.
isClosed()		Indicates that the page has been closed.
isJavaScriptEnabled()		
mainFrame()		
metrics()		
on(eventName, handler)		Listen to page events.
once(eventName, handler)		
pdf(options)		
queryObjects(prototypeHandle)		This method iterates the JavaScript heap and fir all objects with the given prototype.
reload(options)		
screenshot(options)		
select(selector, values)		Triggers a `change` and `input` event once all the provided options have been selected. If there's `<select>` element matching `selector`, the meth throws an error.
setBypassCSP(enabled)		Toggles bypassing page's Content-Security-Poli
setCacheEnabled(enabled)		Toggles ignoring cache for each request based the enabled state. By default, caching is enablec

setContent(html, options)		
setCookie(cookies)		
setDefaultNavigationTimeout(timeout)		This setting will change the default maximum navigation time for the following methods and related shortcuts:- page.goBack(options)- page.goForward(options)- page.goto(url,option page.reload(options)- page.setContent(html,options)- page.waitForNavigation(options)
setDefaultTimeout(timeout)		
setExtraHTTPHeaders(headers)		The extra HTTP headers will be sent with every request the page initiates. NOTE: All HTTP head names are lowercased. (HTTP headers are case-insensitive, so this shouldn't impact your server code.) NOTE: page.setExtraHTTPHeaders does n guarantee the order of headers in the outgoing requests.
setGeolocation(options)		Sets the page's geolocation.
setJavaScriptEnabled(enabled)		
setOfflineMode(enabled)		
setRequestInterception(value)		
setUserAgent(userAgent, userAgentMetadata)		
setViewport(viewport)		`page.setViewport` will resize the page. A lot of websites don't expect phones to change size, sc you should set the viewport before navigating t the page.In the case of multiple pages in a singl browser, each page can have its own viewport s
tap(selector)		This method fetches an element with `selector`, scrolls it into view if needed, and then uses Page.touchscreen to tap in the center of the element. If there's no element matching `selecto` the method throws an error.
target()		
title()		
type(selector, text, options)		Sends a `keydown`, `keypress/input`, and `keyup` eve for each character in the text.To press a special l like `Control` or `ArrowDown`, use Keyboard.press().
url()		
viewport()		

waitFor(selectorOrFunctionOrTimeout, options, args)		
waitForFileChooser(options)		This method is typically coupled with an action triggers file choosing. The following example cli a button that issues a file chooser and then responds with `/tmp/myfile.pdf` as if a user has selected this file.

```
const [fileChooser] = await Promise.all([
page.waitForFileChooser(),
page.click('#upload-file-button'),
// some button that triggers file selection
]);
await fileChooser.accept(['/tmp/myfile.pdf']);
```

NOTE: This must be called before the file chooser is launched. It will not return a currently active file chooser. | | waitForFunction(pageFunction, options, args) | | The `waitForFunction` can be used to observe viewport size change:

```
const puppeteer = require('puppeteer');
(async () => {
const browser = await puppeteer.launch();
const page = await browser.newPage();
const watchDog = page.waitForFunction('window.innerWidth < 100');
await page.setViewport({ width: 50, height: 50 });
await watchDog;
await browser.close();
})();
```

To pass arguments from node.js to the predicate of `page.waitForFunction` function:

```
const selector = '.foo';
await page.waitForFunction(
(selector) => !!document.querySelector(selector),
{},
selector
);
```

The predicate of `page.waitForFunction` can be asynchronous too:

```
const username = 'github-username';
await page.waitForFunction(
async (username) => {
const githubResponse = await fetch(
 `https://api.github.com/users/${username}`
);
const githubUser = await githubResponse.json();
// show the avatar
const img = document.createElement('img');
img.src = githubUser.avatar_url;
```

```
// wait 3 seconds
await new Promise((resolve, reject) => setTimeout(resolve, 3000));
img.remove();
},
{},
username
);
```

| | waitForNavigation(options) | | This resolves when the page navigates to a new URL or reloads. It is useful when you run code that will indirectly cause the page to navigate. Consider this example:

```
const [response] = await Promise.all([
page.waitForNavigation(), // The promise resolves after navigation has finished
page.click('a.my-link'), // Clicking the link will indirectly cause a navigation
]);
```

| | waitForRequest(urlOrPredicate, options) | | | | waitForResponse(urlOrPredicate, options) | | | | waitForSelector(selector, options) | | Wait for the `selector` to appear in page. If at the moment of calling the method the `selector` already exists, the method will return immediately. If the `selector` doesn't appear after the `timeout` milliseconds of waiting, the function will throw.This method works across navigations:

```
const puppeteer = require('puppeteer');
(async () => {
const browser = await puppeteer.launch();
const page = await browser.newPage();
let currentURL;
page
.waitForSelector('img')
.then(() => console.log('First URL with image: ' + currentURL));
for (currentURL of [
'https://example.com',
'https://google.com',
'https://bbc.com',
]) {
await page.goto(currentURL);
}
await browser.close();
})();
```

| | waitForTimeout(milliseconds) | | Causes your script to wait for the given number of milliseconds. | | waitForXPath(xpath, options) | | Wait for the `xpath` to appear in page. If at the moment of calling the method the `xpath` already exists, the method will return immediately. If the `xpath` doesn't appear after the `timeout` milliseconds of waiting, the function will throw.This method works across navigation

```
const puppeteer = require('puppeteer');
(async () => {
const browser = await puppeteer.launch();
const page = await browser.newPage();
let currentURL;
page
```

```
.waitForXPath('//img')
.then(() => console.log('First URL with image: ' + currentURL));
for (currentURL of [
'https://example.com',
'https://google.com',
'https://bbc.com',
]) {
await page.goto(currentURL);
}
await browser.close();
})();
```

| | workers() | | |