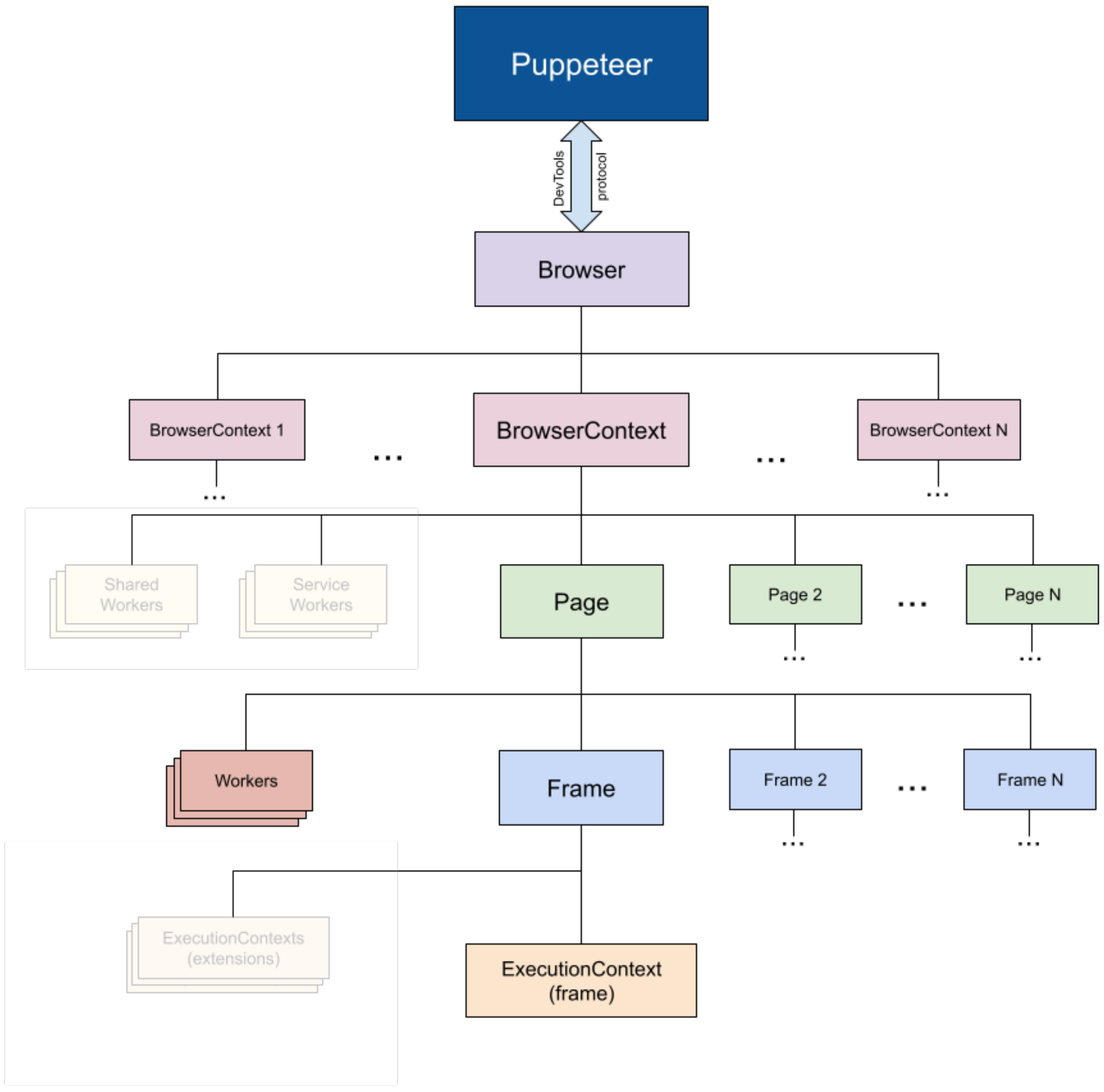
本文主要介绍puppeteer的入门，讲解puppeteer的常见API

puppeteer简介

Puppeteer 是一个 Node 库，它提供了一个高级 API 来通过 DevTools 协议控制 Chromium 或 Chrome。

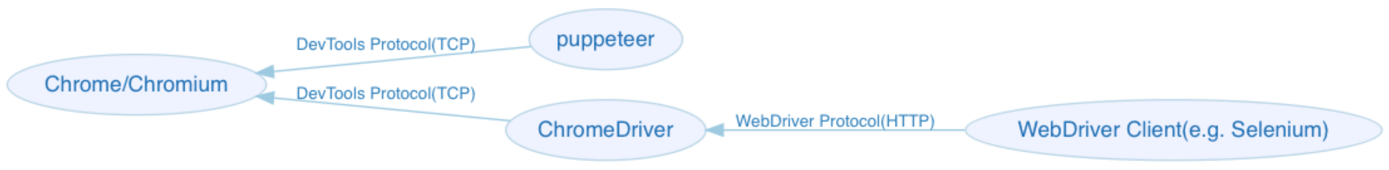
Puppeteer API 是分层次的，反映了浏览器结构。



puppeteer VS selenium的原理区别

puppeteer：直接使用 DevTools Protocol 控制浏览器

selenium：webdriver是对 DevTools Protocol 二次封装后的 Protocol，selenium是一种webdriver客户端



1-创建脚本

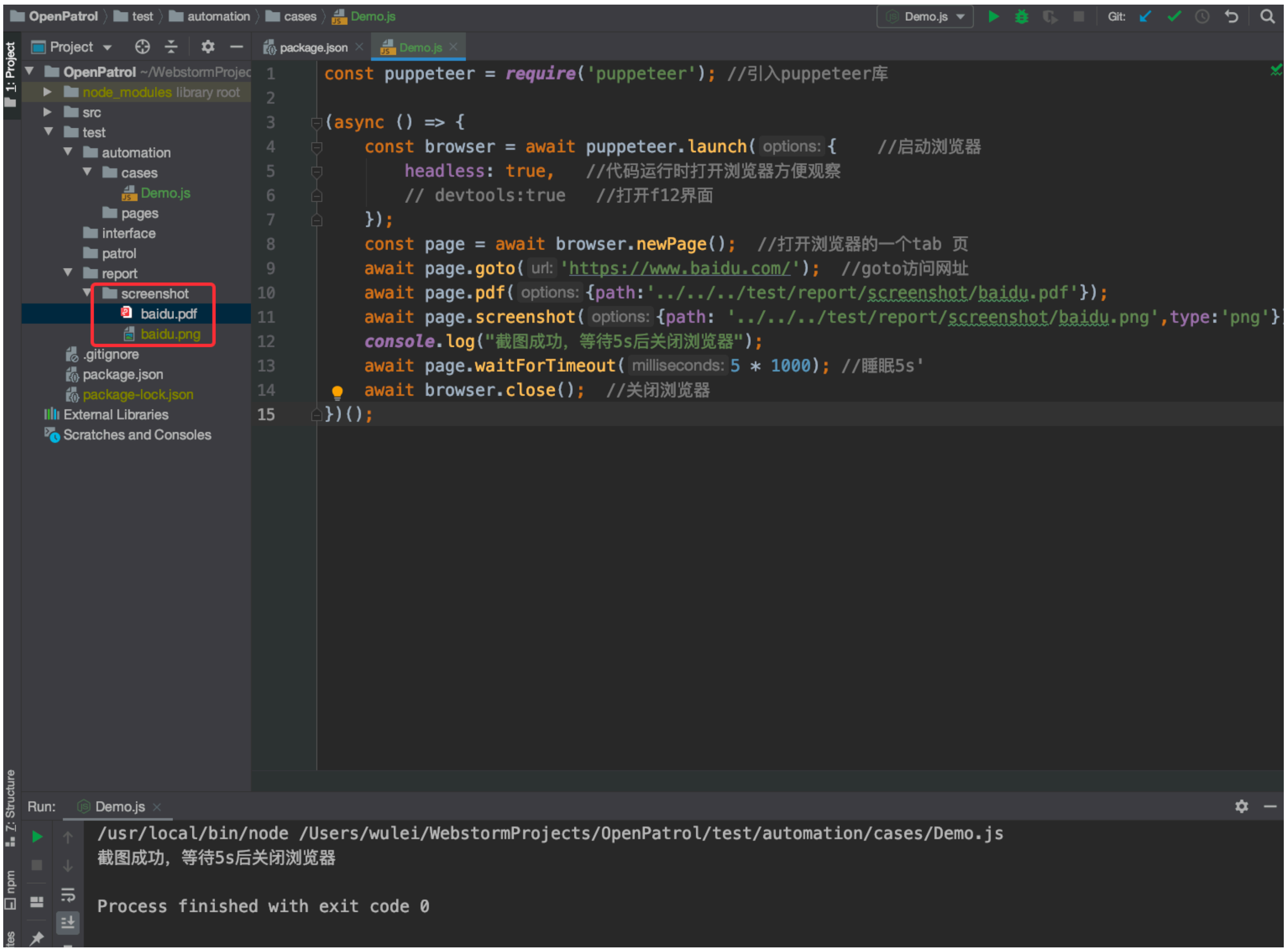
创建 test/automation/cases/Demo.js

```
const puppeteer = require('puppeteer'); //引入puppeteer库

(async () => {
    const browser = await puppeteer.launch({     //启动浏览器
        headless: false,    //代码运行时打开浏览器方便观察
        // devtools:true    //打开f12界面
    });
    const page = await browser.newPage();  //打开浏览器的一个tab 页
    await page.goto('https://www.baidu.com/');  //goto访问网址
    await page.pdf({path:'../../../test/report/screenshot/baidu.pdf'});//PDF
    await page.screenshot({path:
'../../../test/report/screenshot/baidu.png',type:'png'});  //截图
    console.log("截图成功，等待5s后关闭浏览器");
    await page.waitForTimeout(5 * 1000); //睡眠5s'
    await browser.close();  //关闭浏览器
})();
```

2-右键运行

右键运行，可以看到截图/PDF保存成功：



```
const puppeteer = require('puppeteer'); //引入puppeteer库

(async () => {
    const browser = await puppeteer.launch( options: {     //启动浏览器
        headless: true,    //代码运行时打开浏览器方便观察
        // devtools:true    //打开f12界面
    });
    const page = await browser.newPage();  //打开浏览器的一个tab 页
    await page.goto( url: 'https://www.baidu.com/');  //goto访问网址
    await page.pdf( options: {path:'../../../test/report/screenshot/baidu.pdf'});
    await page.screenshot( options: {path: '../../../test/report/screenshot/baidu.png',type:'png'}
    console.log("截图成功, 等待5s后关闭浏览器");
    await page.waitForTimeout( milliseconds: 5 * 1000); //睡眠5s'
    await browser.close();  //关闭浏览器
})();
```

```
/usr/local/bin/node /Users/wulei/WebstormProjects/OpenPatrol/test/automation/cases/Demo.js
截图成功, 等待5s后关闭浏览器

Process finished with exit code 0
```

Puppeteer常见API介绍

在demo例子中，可以看到打开浏览器，跳转链接，截图，等待，关闭浏览器等操作。下面对常见的API操作进行讲解和演示：

完整API参考：https://zhaoqize.github.io/puppeteer-api-zh_CN/#?product=Puppeteer

1-引入依赖

```
const puppeteer = require('puppeteer'); //引入puppeteer库
```

const:是javascript内置的一个关键字，const用于声明一个或多个常量

require:导入以来

2-puppeteer.launch()

功能：指定参数启动并返回浏览器实例

返回：Promise

launch options参数继承：LaunchOptions， BrowserLaunchArgumentOptions， BrowserConnectOptions

```
export declare function launch(options?: LaunchOptions & BrowserLaunchArgumentOptions &
BrowserConnectOptions & {
        product?: Product;
        extraPrefsFirefox?: Record<string, unknown>;
    }): Promise<Browser>;
```

LaunchOptions

```
export declare interface LaunchOptions {
    /**
     * Chrome Release Channel
     */
    channel?: ChromeReleaseChannel;
    /**
     * Path to a browser executable to use instead of the bundled Chromium. Note
     * that Puppeteer is only guaranteed to work with the bundled Chromium, so use
     * this setting at your own risk.
     */
    executablePath?: string;
    /**
     * If `true`, do not use `puppeteer.defaultArgs()` when creating a browser. If
     * an array is provided, these args will be filtered out. Use this with care -
     * you probably want the default arguments Puppeteer uses.
     * @defaultValue false
     */
    ignoreDefaultArgs?: boolean | string[];
    /**
     * Close the browser process on `Ctrl+C`.
     * @defaultValue `true`
     */
    handleSIGINT?: boolean;
    /**
     * Close the browser process on `SIGTERM`.
     * @defaultValue `true`
     */
    handleSIGTERM?: boolean;
    /**
     * Close the browser process on `SIGHUP`.
     * @defaultValue `true`
     */
    handleSIGHUP?: boolean;
    /**
     * Maximum time in milliseconds to wait for the browser to start.
     * Pass `0` to disable the timeout.
     * @defaultValue 30000 (30 seconds).
     */
    timeout?: number;
    /**
     * If true, pipes the browser process stdout and stderr to `process.stdout`
     * and `process.stderr`.
     * @defaultValue false
     */
    dumpio?: boolean;
    /**
     * Specify environment variables that will be visible to the browser.
     * @defaultValue The contents of `process.env`.
     */
    env?: Record<string, string | undefined>;
    /**
     * Connect to a browser over a pipe instead of a WebSocket.
     * @defaultValue false
     */
    pipe?: boolean;
    /**
     * Which browser to launch.
     * @defaultValue `chrome`
     */
    product?: Product;
    /**
     * {@link https://searchfox.org/mozilla-release/source/modules/libpref/init/all.js
| Additional preferences } that can be passed when launching with Firefox.
     */
    extraPrefsFirefox?: Record<string, unknown>;
    /**
     * Whether to wait for the initial page to be ready.
     * Useful when a user explicitly disables that (e.g. `--no-startup-window` for
Chrome).
     * @defaultValue true
     */
    waitForInitialPage?: boolean;
}
```

BrowserLaunchArgumentOptions

```
export declare interface BrowserLaunchArgumentOptions {
    /**
     * Whether to run the browser in headless mode.
     * @defaultValue true
     */
    headless?: boolean | 'chrome';
    /**
     * Path to a user data directory.
     * {@link
https://chromium.googlesource.com/chromium/src/+/refs/heads/main/docs/user_data_dir.md
| see the Chromium docs}
     * for more info.
     */
    userDataDir?: string;
    /**
     * Whether to auto-open a DevTools panel for each tab. If this is set to
     * `true`, then `headless` will be forced to `false`.
     * @defaultValue `false`
     */
    devtools?: boolean;
    /**
     *
     */
    debuggingPort?: number;
    /**
     * Additional command line arguments to pass to the browser instance.
     */
    args?: string[];
}
```

BrowserConnectOptions

```
export declare interface BrowserConnectOptions {
    /**
     * Whether to ignore HTTPS errors during navigation.
     * @defaultValue false
     */
    ignoreHTTPSErrors?: boolean;
    /**
     * Sets the viewport for each page.
     */
    defaultViewport?: Viewport | null;
    /**
     * Slows down Puppeteer operations by the specified amount of milliseconds to
     * aid debugging.
     */
    slowMo?: number;
    /**
     * Callback to decide if Puppeteer should connect to a given target or not.
     */
    targetFilter?: TargetFilterCallback;
}
```

3-Browser

当 Puppeteer 连接到一个 Chromium 实例的时候会通过 `puppeteer.launch` 或 `puppeteer.connect` 创建一个 Browser 对象。

最常见的用法：

方法	功能说明
browser.newPage()	返回一个新的 Page 对象。Page 在一个默认的浏览器上下文中被创建。
browser.close()	关闭 Chromium 及其所有页面(如果页面被打开的话)
browser.targets()	浏览器内所有活动目标组成的数组。在多个浏览器上下文的情况下，该方法将返回一个包含所有浏览器上下文中的所有目标的数组。多用于遍历。
browser.pages()	返回一个浏览器中所有页面的数组。 在多个浏览器上下文的情况下， 该方法将返回一个包含所有浏览器上下文中所有页面的数组。多用于遍历。

4-Page

Page 提供了操作页面的方法，是最重要的也是后面使用最多的。下面简单介绍一下最常用的方法。

page.on():浏览器事件监听

功能：Page会触发多种事件（下面描述的），可以用 `node` 原生的方法 来捕获处理，比如 `on`,`once` 或者 `removeListener`。后续讲解UI自动化时会重点讲解。

- page.on('close')v0.9.0
- page.on('console')v0.9.0
- page.on('dialog')v0.9.0
- page.on('domcontentloaded')v0.9.0
- page.on('error')v0.9.0
- page.on('frameattached')v0.9.0
- page.on('framedetached')v0.9.0
- page.on('framenavigated')v0.9.0
- page.on('load')v0.9.0
- page.on('metrics')v0.9.0
- page.on('pageerror')v0.9.0
- page.on('request')v0.9.0
- page.on('requestfailed')v0.9.0
- page.on('requestfinished')v0.9.0
- page.on('response')v0.9.0
- page.on('workercreated')v0.9.0
- page.on('workerdestroyed')v0.9.0

page.goto():打开页面

功能：打开指定页面

```
goto(url: string, options?: WaitForOptions & {
    referer?: string;
}): Promise<HTTPResponse>;
```

WaitForOptions:

```
export declare interface WaitForOptions {
    timeout?: number;
    waitUntil?: PuppeteerLifeCycleEvent | PuppeteerLifeCycleEvent[];
}
```

page.pdf():保存PDF

功能：按照参数对页面保存为PDF

PDFOptions：

```
export declare interface PDFOptions {
    /**
     * Scales the rendering of the web page. Amount must be between `0.1` and `2`.
     * @defaultValue 1
     */
    scale?: number;
    /**
     * Whether to show the header and footer.
     * @defaultValue false
     */
    displayHeaderFooter?: boolean;
    /**
     * HTML template for the print header. Should be valid HTML with the following
     * classes used to inject values into them:
     * - `date` formatted print date
     *
     * - `title` document title
     *
     * - `url` document location
     *
     * - `pageNumber` current page number
     *
     * - `totalPages` total pages in the document
     */
    headerTemplate?: string;
    /**
     * HTML template for the print footer. Has the same constraints and support
     * for special classes as {@link PDFOptions.headerTemplate}.
     */
    footerTemplate?: string;
    /**
     * Set to `true` to print background graphics.
     * @defaultValue false
     */
    printBackground?: boolean;
    /**
     * Whether to print in landscape orientation.
     * @defaultValue = false
     */
    landscape?: boolean;
    /**
     * Paper ranges to print, e.g. `1-5, 8, 11-13`.
     * @defaultValue The empty string, which means all pages are printed.
     */
    pageRanges?: string;
    /**
     * @remarks
     * If set, this takes priority over the `width` and `height` options.
     * @defaultValue `letter`.
     */
    format?: PaperFormat;
    /**
     * Sets the width of paper. You can pass in a number or a string with a unit.
     */
    width?: string | number;
    /**
     * Sets the height of paper. You can pass in a number or a string with a unit.
     */
    height?: string | number;
    /**
     * Give any CSS `@page` size declared in the page priority over what is
     * declared in the `width` or `height` or `format` option.
     * @defaultValue `false`, which will scale the content to fit the paper size.
     */
    preferCSSPageSize?: boolean;
    /**
     * Set the PDF margins.
     * @defaultValue no margins are set.
     */
    margin?: PDFMargin;
    /**
     * The path to save the file to.
     *
     * @remarks
     *
     * If the path is relative, it's resolved relative to the current working
directory.
     *
     * @defaultValue the empty string, which means the PDF will not be written to disk.
     */
    path?: string;
    /**
     * Hides default white background and allows generating pdfs with transparency.
     * @defaultValue false
     */
    omitBackground?: boolean;
    /**
```

```
     * Timeout in milliseconds
     * @defaultValue 30000
     */
    timeout?: number;
}
```

参数解释:

```
path <string> pdf文件保存的路径。如果是相对路径，则相对当前路径。如果不指定路径，将不保存到硬盘。
scale <number> 页面渲染的缩放。默认是1。缩放值必须介于0.1到2之间。
displayHeaderFooter <boolean> 显示页眉和页脚。默认是不显示
headerTemplate <string> 页眉的html模板，可以有这些变量:
date 格式化的日期
title 网页标题
url 网页地址
pageNumber 当前页码
totalPages 总页数
footerTemplate <string> 页脚的html模板。和页眉模板变量相同。
printBackground <boolean> 是否打印背景图. 默认是 false。
landscape <boolean> 页面横向(?Paper orientation). 默认为 false.
pageRanges <string> 要输出的页码范围, 比如, '1-5, 8, 11-13'。默认是空字符串, 表示全部页码。
format <string> 页面格式。如果设置了, 将覆盖 width 和 height 配置. 默认是 'Letter'。
width <string> 页面宽度, 接受带单位的字符串。
height <string> 页面高度, 接受带单位的字符串。
margin <Object> 页面空白白边配置, 默认是空
top <string> 顶部的白边
right <string> 右侧白边, 接受带单位的字符串
bottom <string> 底部白边, 接受带单位的字符串
left <string> 左侧白边, 接受带单位的字符串
preferCSSPageSize <boolean> 给页面优先级声明的任何CSS @page 大小超过 width 和 height 或
format 选项中声明的大小。 默认为 false, 它将缩放内容以适合纸张大小。
```

page.screenshot():截图

功能：按照参数对页面截图保存

```
screenshot(options?: ScreenshotOptions): Promise<Buffer | string>;
```

ScreenshotOptions:

```
export declare interface ScreenshotOptions {
    /**
     * @defaultValue 'png'
     */
    type?: 'png' | 'jpeg' | 'webp';
    /**
     * The file path to save the image to. The screenshot type will be inferred
     * from file extension. If path is a relative path, then it is resolved
     * relative to current working directory. If no path is provided, the image
     * won't be saved to the disk.
     */
    path?: string;
    /**
     * When true, takes a screenshot of the full page.
     * @defaultValue false
     */
    fullPage?: boolean;
    /**
     * An object which specifies the clipping region of the page.
     */
    clip?: ScreenshotClip;
    /**
     * Quality of the image, between 0-100. Not applicable to `png` images.
     */
    quality?: number;
    /**
     * Hides default white background and allows capturing screenshots with
transparency.
     * @defaultValue false
     */
    omitBackground?: boolean;
    /**
     * Encoding of the image.
     * @defaultValue 'binary'
     */
    encoding?: 'base64' | 'binary';
    /**
     * If you need a screenshot bigger than the Viewport
     * @defaultValue true
     */
    captureBeyondViewport?: boolean;
}
```

page.waitForTimeout():等待

功能：等待ms

page后续其他的常见API会在UI自动化，巡检等章节展开