Сервис hh.ru позволяет использовать только изображения для портфолио. Ссылки на репозитории модулей, представленных далее, и адрес аккаунта на github указаны в начале пункта «О себе» резюме.

Портфолио

`browserControl.js`

репозиторий: github.com/ArundhatiApte/browser-control-js[1]
документация на русском: github.com/ArundhatiApte/browser-control-js/tree/main/doc[2]

browserControl — модуль для управления браузером, предоставляющий интерфейсы из WebExtensions, имеющий функциональность схожую с предоставляемой Selenuim и Puppeter. Следующий фрагмент кода запускает веб-обозреватель waterfox, открывает страницу example.com в новой вкладке и выполняет в ней некие действия:

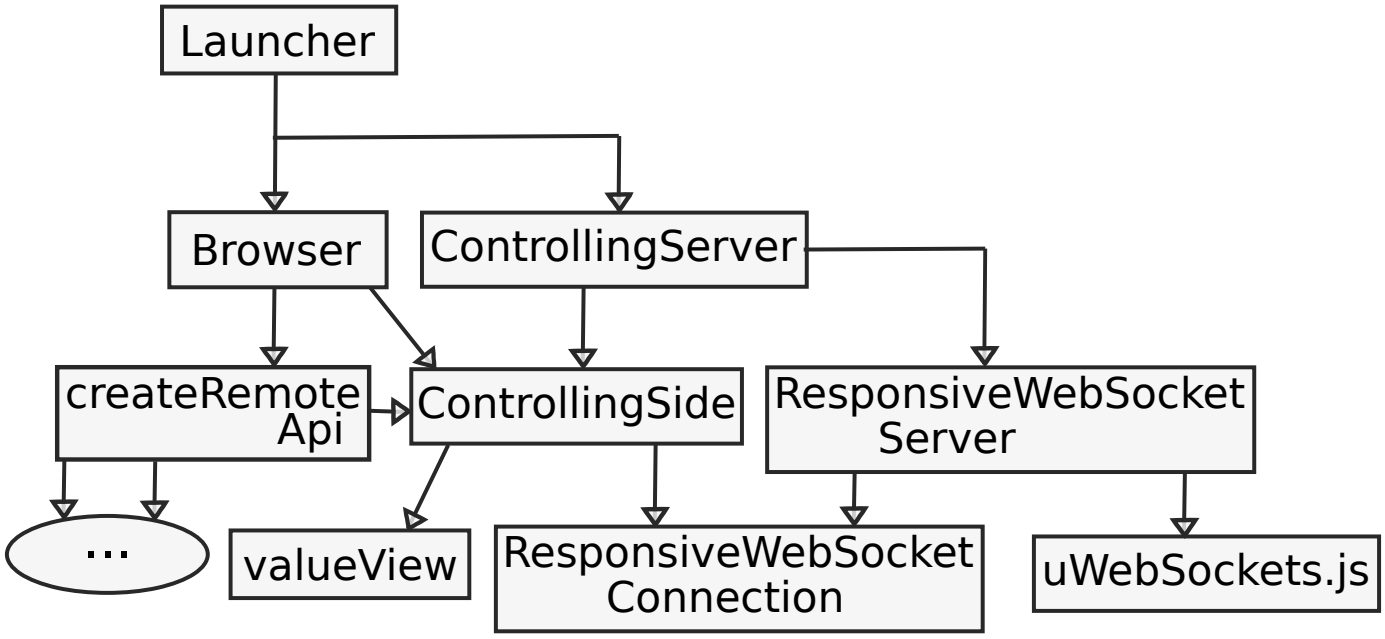
```
import launcher from "browserControl"

(async function main() {
  const browser = await launcher.launch("waterfox")
  const tabs = browser.tabs
  const tab = await tabs.create({ url: "https://example.com" })
  const resultFromFrames = await tabs.executeScript(tab.id, {
    code: "console.log(\"сделать что нибудь\");window.screen.height"
  })
  // …
 })()
```

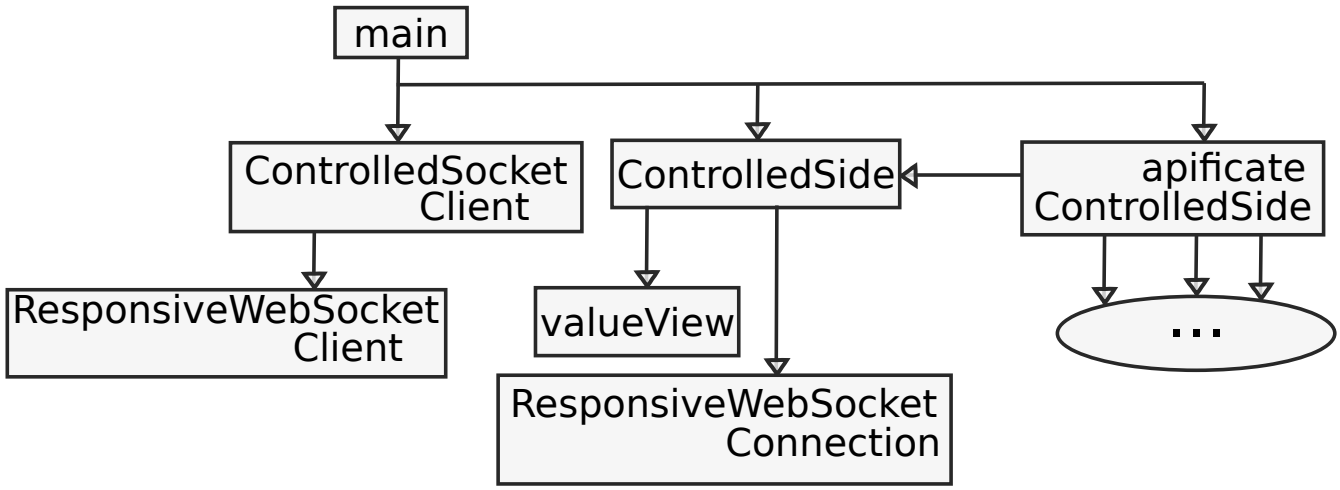
Модуль состоит из клиентской части — расширения браузера, экспортирующего интерфейсы из WebExtenstions, принимающего запросы на вызов методов и отправляющего сообщения о событиях. Серверная часть — npm пакет, предоставляющего класс Browser, который внутренне посылает запросы на вызов методов и слушает сообщения о событиях и создаётся динамически на основе полученной структуры, описывающей клиентский API.

Для обмена данными используются WebSockets. На их основе создан подмодуль ResponsiveWebSockets, позволяющий отправлять запросы и незапрашивающие сообщения через данный протокол. Классы контролирующего соединения — ControllingConnection, и контролируемого — ControlledConnection, используют отзывчивые WebSockets при композиции. В версии browserControl на JavaScript применена динамическая природа данного языка — на стороне сервера API класса Browser строится на лету из полученного описания интерфейса. Функционал, экспортируемый клиентом, имеет возможность изменения и расширения, подробности последего описаны в этой статье[3].

Серверная сторона



Клиентская сторона



В общем применялся объектно-ориентированный и функциональный стиль. Из шаблонов проектирования GoF использованы: фабричный метод, медиатор, шаблонный метод. Для созданий отдельный классов на стороне клиента было применено встраивание зависимостей — DI, что дало возможность запуска тестов в node.js за счёт предоставления поддельный объектов. Из модулей 3-их сторон использовал uWebSockets.js и ws. Следующие инструменты были задействованы при разработке:

- jshint — статическое обследование кода, линтинг
- mocha — тестирование
- nyc — определение уровня покрытия тестами
- code-complexity — измерение цикломатической сложности кода
- webpack и bash — сценарий, собирающий расширение браузера

Метрики кода на JavaScript:

- кол-во строк основного кода, учитывая пустые строки: 4373
- кол-во строк тестирующего кода, учитывая пустые строки: 5340
- средняя/медианная цикломатическая сложность кода: 12.63/10
- максимальная цикломатическая сложность кода: 53.33

`ResponsiveWebSockets`

репозитории:

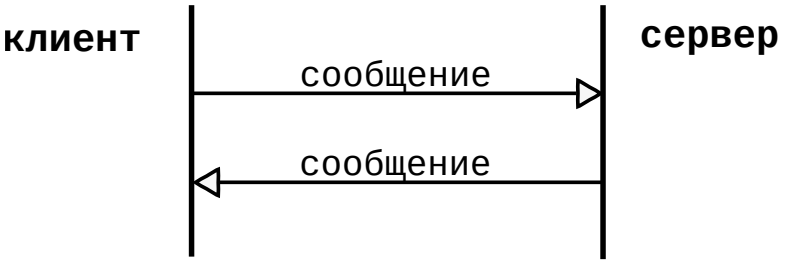
- версия на JavaScript: github.com/ArundhatiApte/responsive-web-sockets-js[4]

  - документация на русском[5]
- версия на Java: github.com/ArundhatiApte/responsive-web-sockets[6]
  - документация на русском[7]

RFC6455 или по-другому WebSocket предназначен для быстрого двунаправленного обмена сообщениями между сервером и клиентом. Данная цель достигается за счёт применения легковесного формата заголовков передаваемых фреймов, чеё размер более чем в 10 раз меньше чем используемый в HTTP, что снижает вычислительные затраты при разборе входящих пакетов.

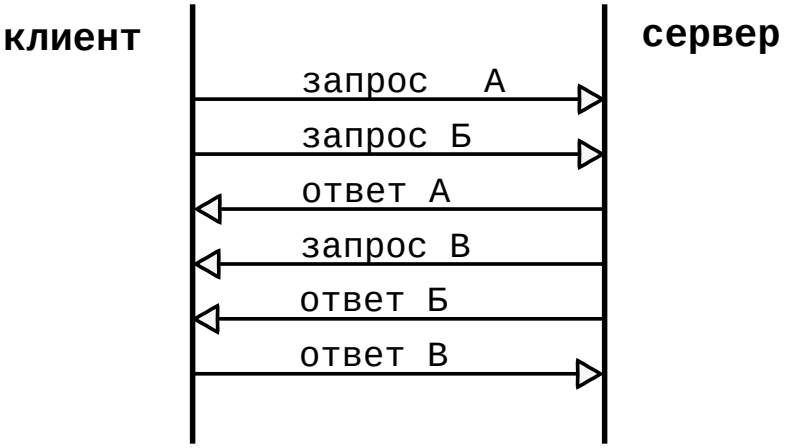
WebSocket соединение имеет метод для отправки сообщений и событие, возникающие при их получении от другой стороны. ResponsiveWebSockets обёртывают протокол RFC6455, добавляя возможность передачи запроса.

Сообщения WebSockets



Запрос/ответ HTTP

Запрос/ответ ResponsiveWebSockets



ResponsiveWebSockets также могут отправлять более 1-го запроса за раз.

Данный модуль изначально был использован при создании вышеописанной системы browserControl для передачи запросов на вызов методов и отправки сообщений о интерактивных событиях. При реализации отзывчивых WebSockets на Java был использован интсрумент сборки и управления зависимостями Maven, на Scala при помощи scalatest написан тестирующий код. Для разработки данного модуля на JavaScript такие же утилиты, упомянутые при описании browserControl.

Расширение браузера переводчика Яндекс

репозиторий: github.com/ArundhatiApte/yandex-translator-browser-extension[8]

Некоммерческое дополнение для браузера, предоставляющее возможность перевода страниц с помощью веб-службы Yandex translate. Расширение протестировано в браузере из семейста firefox, проект находиться на ранней стадии. Следующими по плану пунктами

являются создание дизайна интерфейса пользователя и доработка поддержки цветовых схем (тёмной и светлой).

Применялся объектно-ориентированный и функциональный стиль. Из шаблонов проектирования GoF были использованы: фабричный метод, адаптер, композит, медиатор и шаблонный метод. Для избегании оплаты запросов к реальному сервису создан сервер, реализующий REST интерфейс веб-службы Yandex Translator, использующий 3-и класса:

- поддельный переводчик, преобразующий текст в верхний регистр
- «приниматель» запросов
- медиатор, организующий совместную работу частей

Задействовано внедрение зависимостей, что дало возможность запуска тестов в node.js модулей, полагающихся на API браузера. Инструменты разработки практически такие же, что были применены в browserControl. Исключением является замена bash на python в сценарии, собирающем дополнение.

Метрики кода на JavaScript:

- кол-во строк основного кода, учитывая пустые строки: 3274
- кол-во строк тестирующего кода, учитывая пустые строки: 5697
- средняя/медианная цикломатическая сложность кода: 19.96/13.953
- максимальная цикломатическая сложность кода: 100

Кратко о других модулях

`TorControls`

репозиторий: github.com/ArundhatiApte/tor-controls-jsgithub.com/ArundhatiApte/tor-controls-js[9]

Компонент на JavaScript для управления сервисом TOR — сети интернет узлов для анонимизации трафика. Данный модуль был создан в результате рефакторинга npm пакета tor-control.

`CountriesCodes`

Репозиторий: github.com/ArundhatiApte/countries-codes[10]

Модуль, представляющий таблице стран, их числовых и символьных кодов, описанную в стандарте ISO3166.

Одна из причин создания данного пакета заключалась в уменьшении потребления оперативной памяти за счёт избегания использования нескольких объектов строк, хранящих одинаковый код страны. Другой повод был связан с избавлением от потребности создания подобной таблицы при разработке баз данных.

`LuhnNumbers`

репозиторий: github.com/ArundhatiApte/luhn-numbers[11]

Алгоритм Луна обычно используется для первичной проверки номеров банковских карт. Также известны другие области его применения. Данный модуль на Java предоставляет функцию, проверяющую номер по алгоритму Луна, и методы, создающие валидные последовательности цифр.