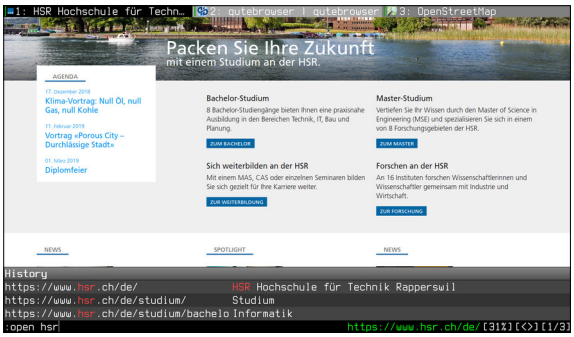
Florian
Bruhin

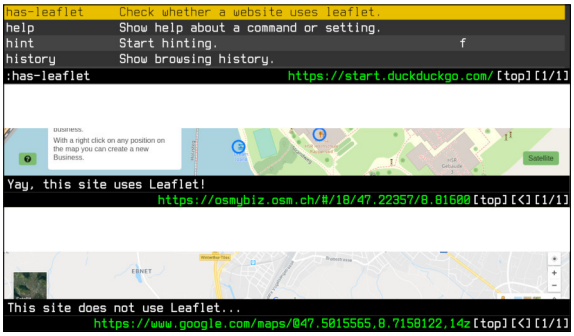
Student	Florian Bruhin
Examiner	Prof. Stefan F. Keller
Subject Area	Software

qutebrowser made extensible



Screenshot of qutebrowser displaying its history completion

Technologies used



Example extension checking whether a website uses the Leaflet library, which is often used to display OpenStreetMap maps

Introduction: The qutebrowser project is a web browser, comparable to Google Chrome or Mozilla Firefox, which is focused on being keyboard-driven and having a minimal user interface. It is aimed at power-users who value customizability and efficiency, but are willing to accept a rather steep learning curve compared to "traditional" web browsers.

Since qutebrowser uses the Python programming language in conjunction with the Qt library for graphical user interfaces, it is standing on the shoulders of giants: It does not implement complex tasks such as downloading and executing HTML/CSS/JavaScript code itself. Instead, it relies on the QtWebEngine project to do so, which is largely based on the same code as Chrome.

Work on qutebrowser started in 2013. Since then, it gained a big community of thousands of users and dozens of contributors.

Objective: In contrast to Chrome and Firefox, qutebrowser does not support extending its functionality via extensions. Over its lifetime, various features have been added to its core by its maintainer and contributors. However, this caused its core to grow substantially, becoming more and more complex over time.

Many users of qutebrowser are power-users and thus have specific feature requests and workflows. It should be possible for those users to extend qutebrowser in an easy way in order to keep qutebrowser's core small and simple.

The goal of this project was to make qutebrowser extensible by introducing a clearly defined API which can be used to develop extensions.

Procedure / Result: Before attempting to expose an interface for extensions from qutebrowser, various problematic areas in its codebase had to be cleaned up due to "technical debt" accumulating in the past. Subsequently, functionality suitable for moving out of the core into extensions was identified. Based on the selected areas of code, a concept for an extension API was developed. After implementing said API, large parts of functionality could be moved into extensions. The resulting changes increased code maintainability and simplicity.

In order to further follow best practices in the software development world, tools for checking data types (thus reducing the chance of accidental software defects) were evaluated. The [i]mypy[/i] tool is now run regularly over the code which resulted in various lingering defects being found in qutebrowser itself and in related projects. Important parts of qutebrowser were annotated with type information in order to find such issues, also serving as additional developer documentation.