Web Browser Automation

Interacting with Rendered Web Pages for Tests and More

Jeremy Bowman
jbowman@edx.org

Why Automate a Browser?

- Verify in-browser JavaScript
- Check for breakage of core functionality
- Confirm compatibility with supported browsers

When Not to Automate a Browser

- Unit tests
- Rendered markup tests
- Data retrieval or web scraping

How to Automate a Browser

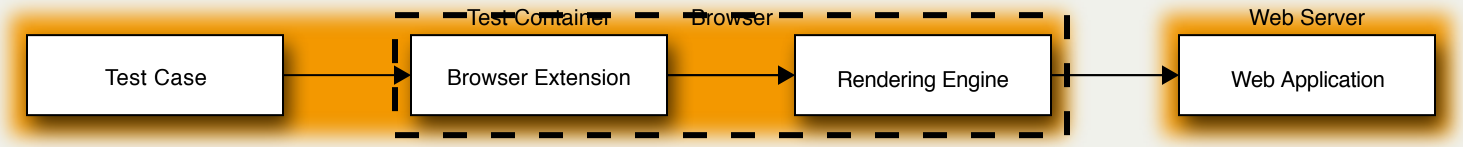
- Selenium
- W3C WebDriver API
- Custom headless browser (PhantomJS, CasperJS, SlimerJS, etc.)

What Is Selenium?

- Browser automation API
- Java server for controlling 1+ browsers
- Client libraries for working with both of the above

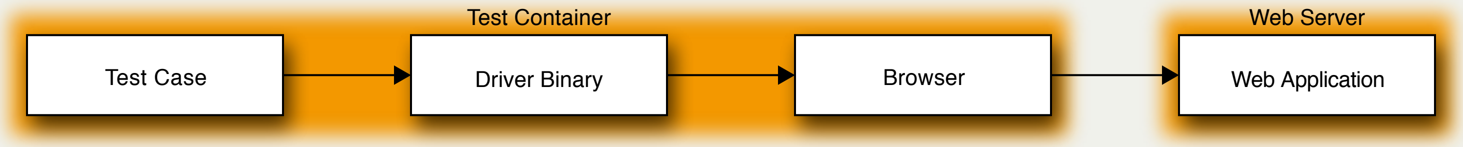
1: Directly Drive a Local Browser

- Firefox 47 and earlier
- Safari

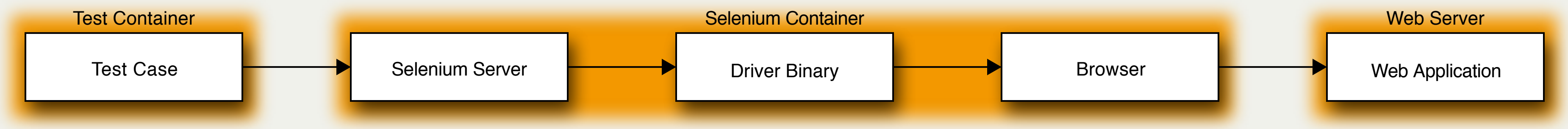


2: Use a Local Binary Driver

- Chrome
- Firefox 47+



3: Drive a Remote Browser



Which Operations Can Be Automated?

- Load a URL
- Find an element
- Get an element’s attributes
- Click an element
- Run a script
- …

Why Not Just Use Selenium Directly?

- Low level operations
- API not a good match for reliable tests
- A minor change in site behavior can impact many tests

Layers Above Selenium

- bok-choy (acceptance tests)
- lettuce (other acceptance tests)
- karma + jasmine (JS unit tests)
- pa11ycrawler (a11y audits)

Thank you!

Questions?

Jeremy Bowman
jbowman@edx.org