Weboberflächen testen

Sind heutige Techniken adäquat?

Claas Thiele
manager methods & processes

it-function software gmbh
Möckernstr. 67, 10965 Berlin

Project Lead

http://molyb.org
http://sf.net/projects/molybdenum

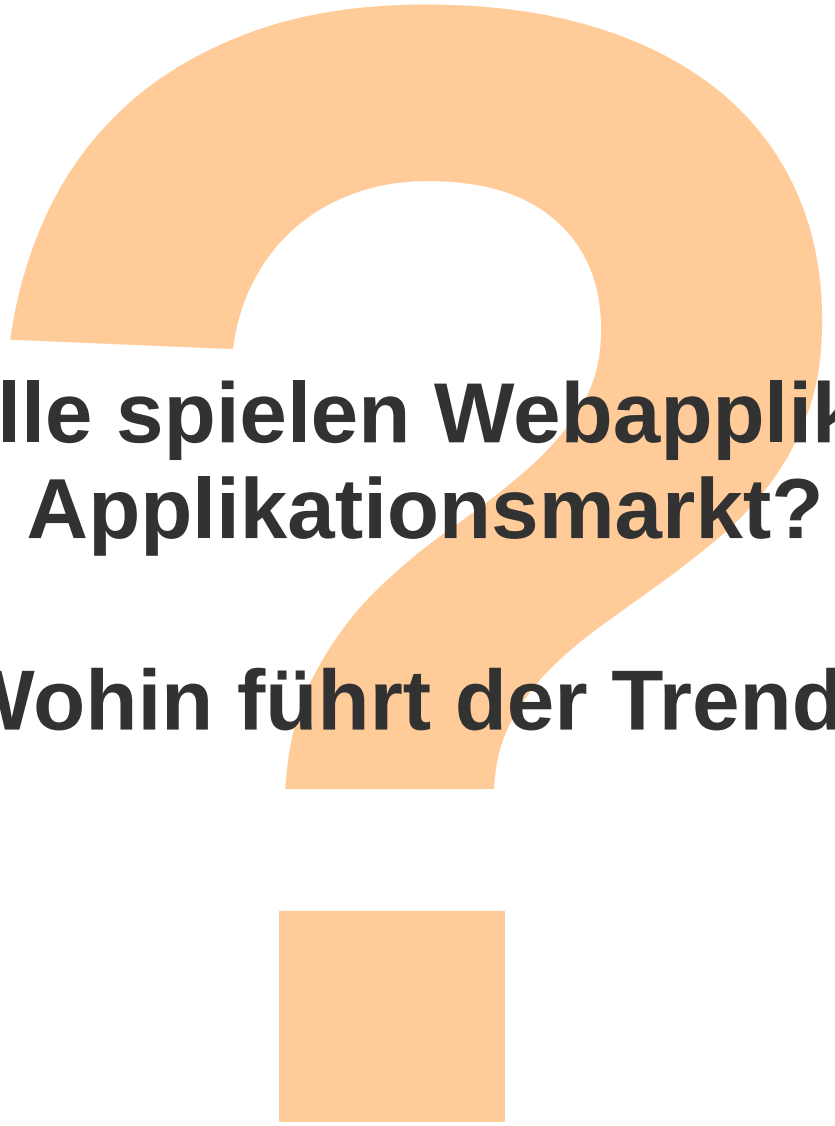
Weboberflächen testen

Sind heutige Techniken adäquat?

Web UI Tests ohne Programmieren

OpenSource (Apache2 Lizenz)
Seit über 5 Jahren aktiv

unterstützt durch



Welche Rolle spielen Webapplikationen im Applikationsmarkt?

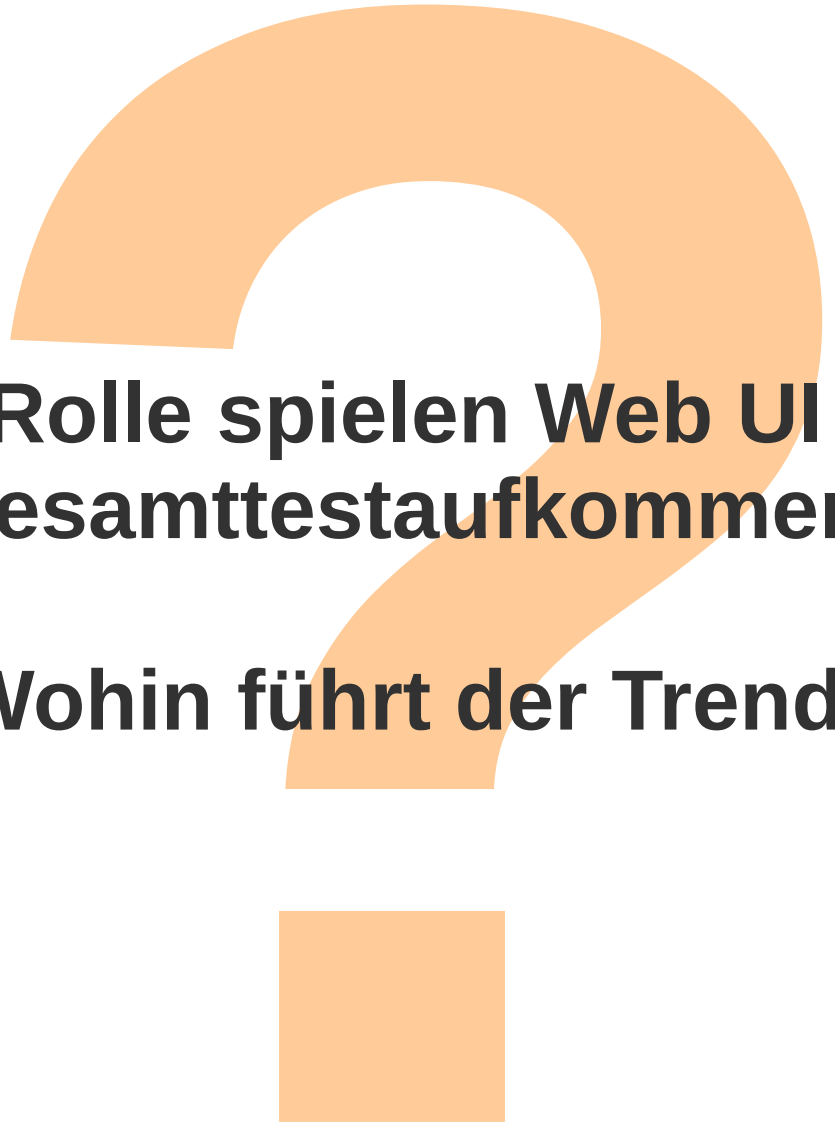
Wohin führt der Trend?

Bedeutung Web UI Tests

Anwendungen wandern ins Web

- Signifikante Masse an alternativen Desktop Systemen
- Bedeutung mobiler Endgeräte nimmt dramatisch zu
- Basistechnologien werden verfügbar
  - UI Bibliotheken
  - HTML5 mit WebM, WebGL, WebSockets und Canvas
  - konkurrenzfähige Javascript Geschwindigkeit)

http://en.wikipedia.org/wiki/Usage_share_of_desktop_operating_systems



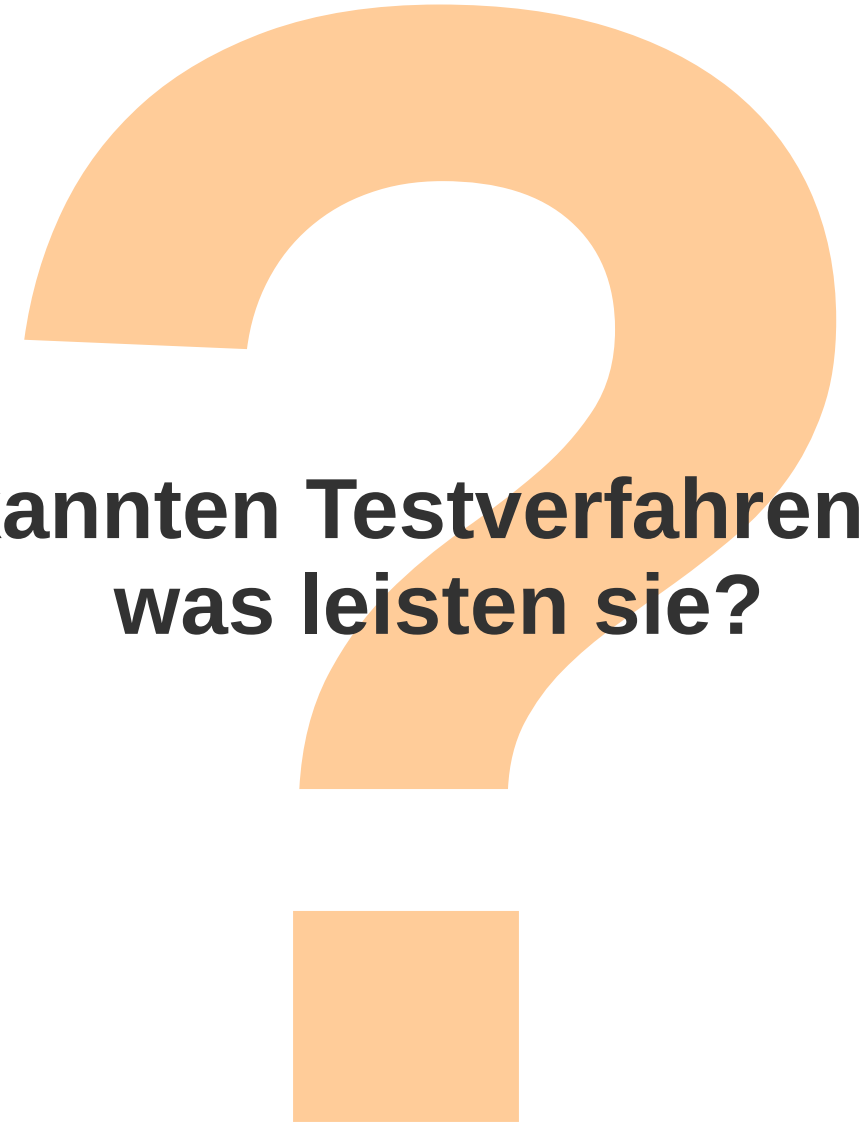
Welche Rolle spielen Web UI Tests im Gesamttestaufkommen?

Wohin führt der Trend?

Bedeutung Web UI Tests

Risiken der Web UI Komponente

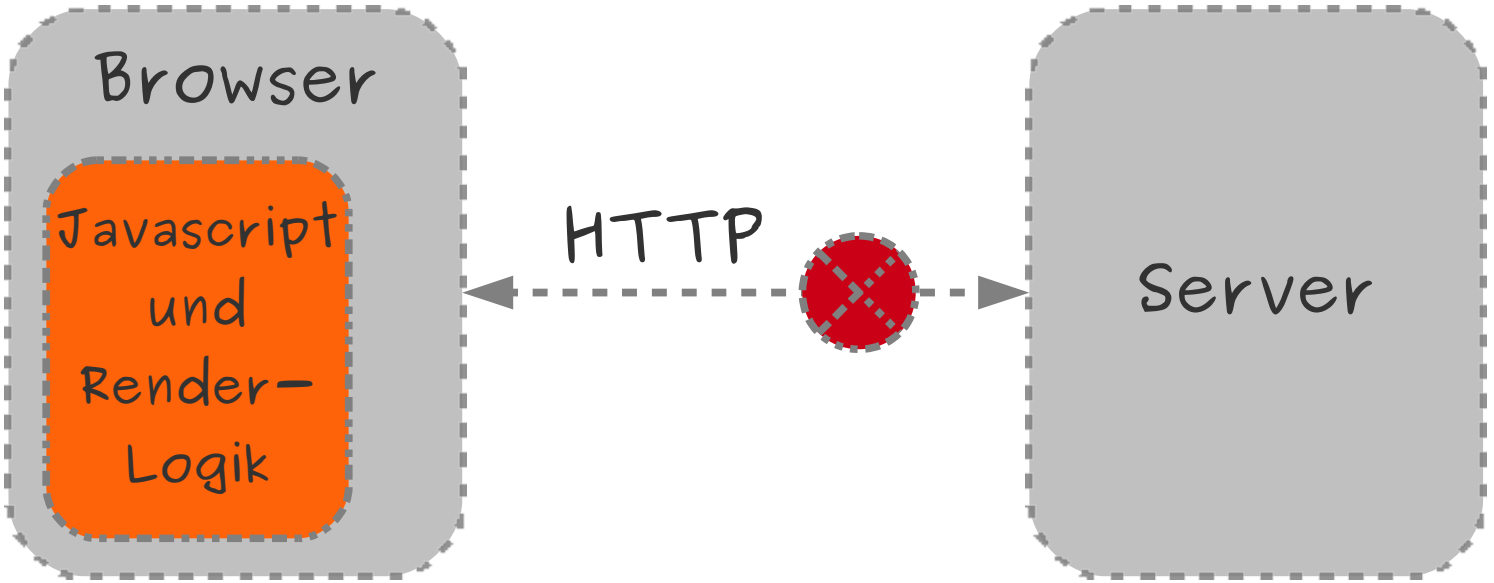
- Komplexität verschiebt sich in den Web Layer
- UI ist Systemgrenze –
  Interface partizipiert an mehreren Tests
- UI ist Ansatzpunkt für Akzeptanztests -
  grössere Bedeutung in Agilen Methoden
- Browser befindet sich in einer 'untrusted domain' -
  sicherheitsrelevante Prüfungen wesentlich
- Browser als Ausführungsumgebung schwer
  kontrollierbar



Welche bekannten Testverfahren gibt es und was leisten sie?

Klassifizierung Testmethoden

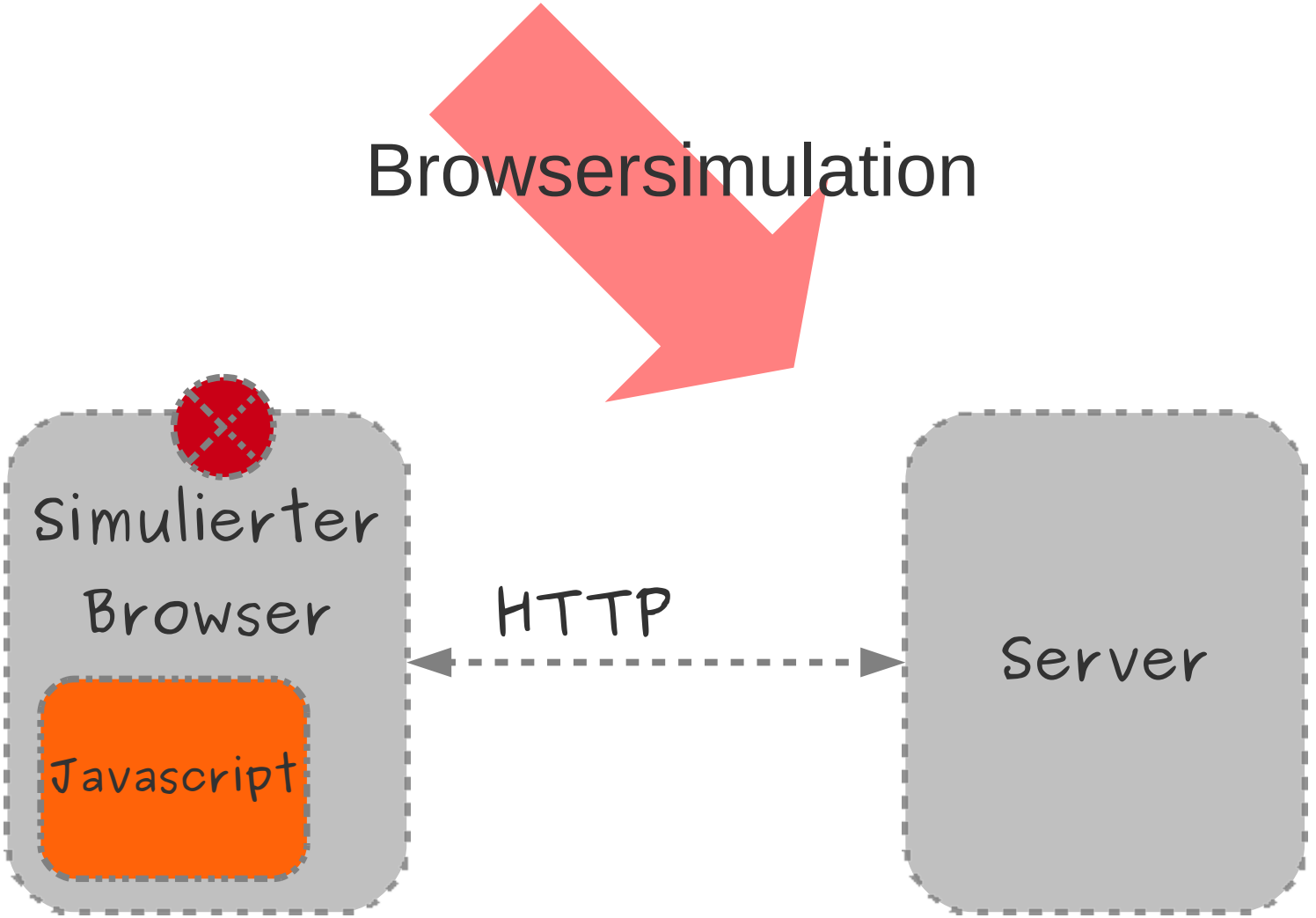
HTTP-Protokoll basiert



HTTP-Protokoll basiert

- Javascript nicht testbar
- Renderergebnis nicht testbar
- Browserunterschiede nur teilweise erfassbar

Klassifizierung Testmethoden



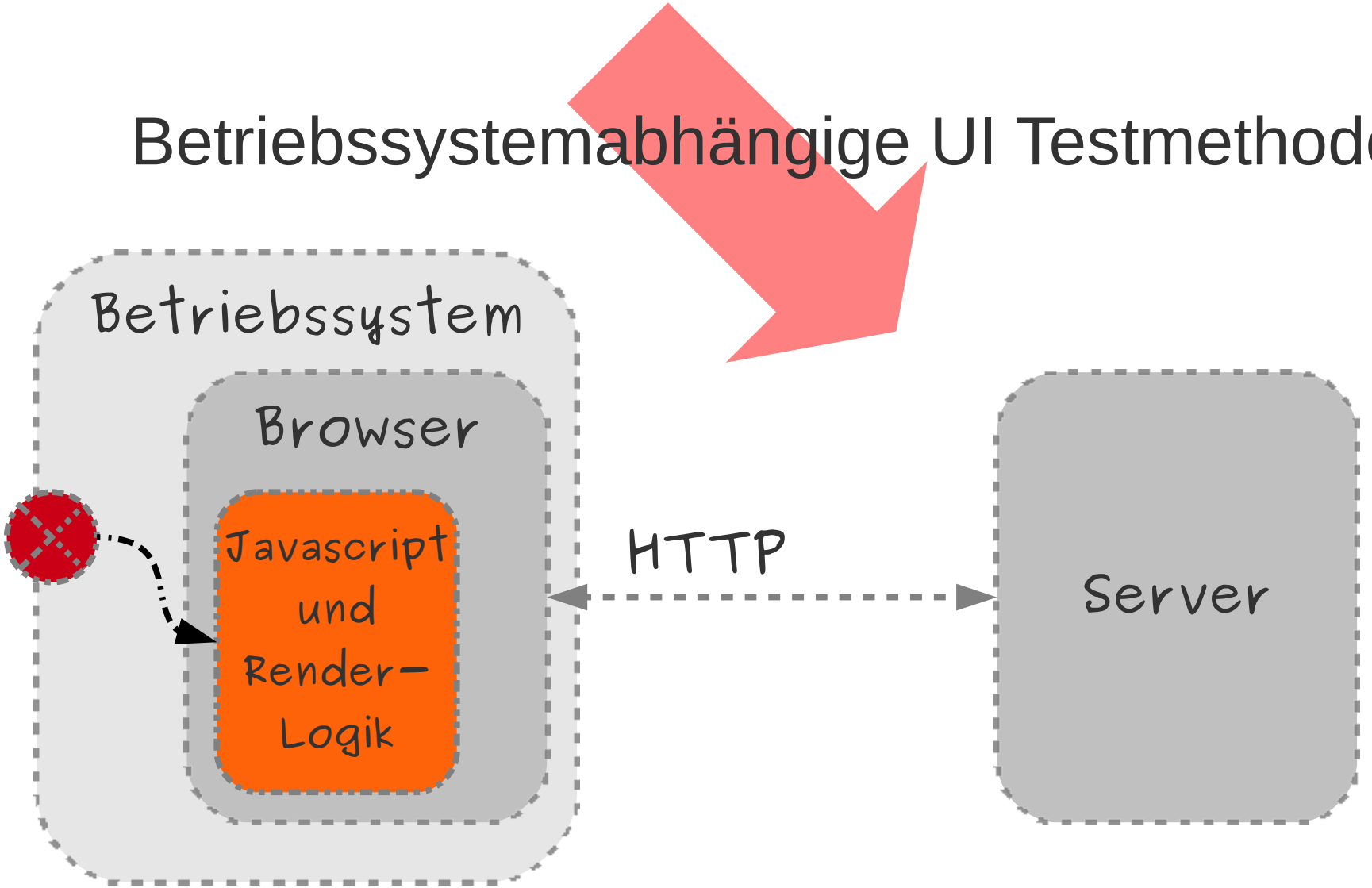
HTMLUnit, Canoo Webtest

Browsersimulation

- Javascript eingeschränkt testbar
- Renderergebnis nicht testbar
- Browserunterschiede nicht erfassbar

Klassifizierung Testmethoden

Betriebssystemabhängige UI Testmethoden



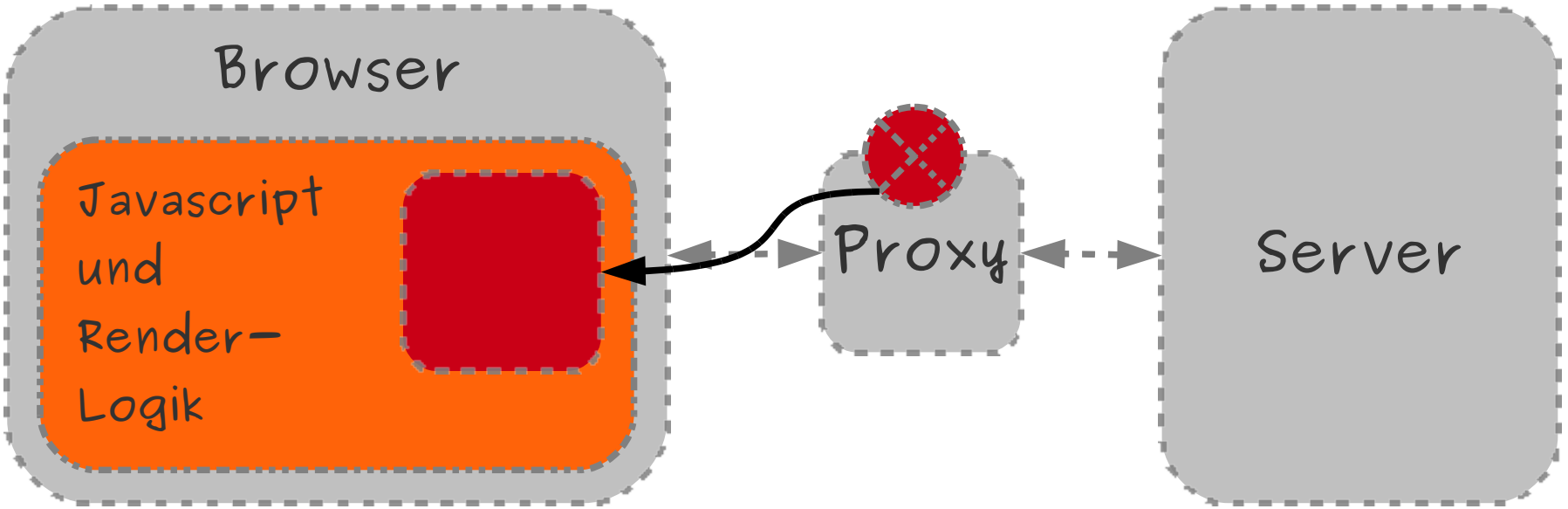
Silktest, Rational Robot, WinRunner

Betriebssystemabhängige UI Testmethoden

- Javascript toolabhängig testbar
- Renderergebnis toolabhängig testbar
- Browserunterschiede toolabhängig erfassbar
- Testprogrammierung toolabhängig
- Aufwendig in Herstellung - teuer

Klassifizierung Testmethoden

DOM basiert – Proxy



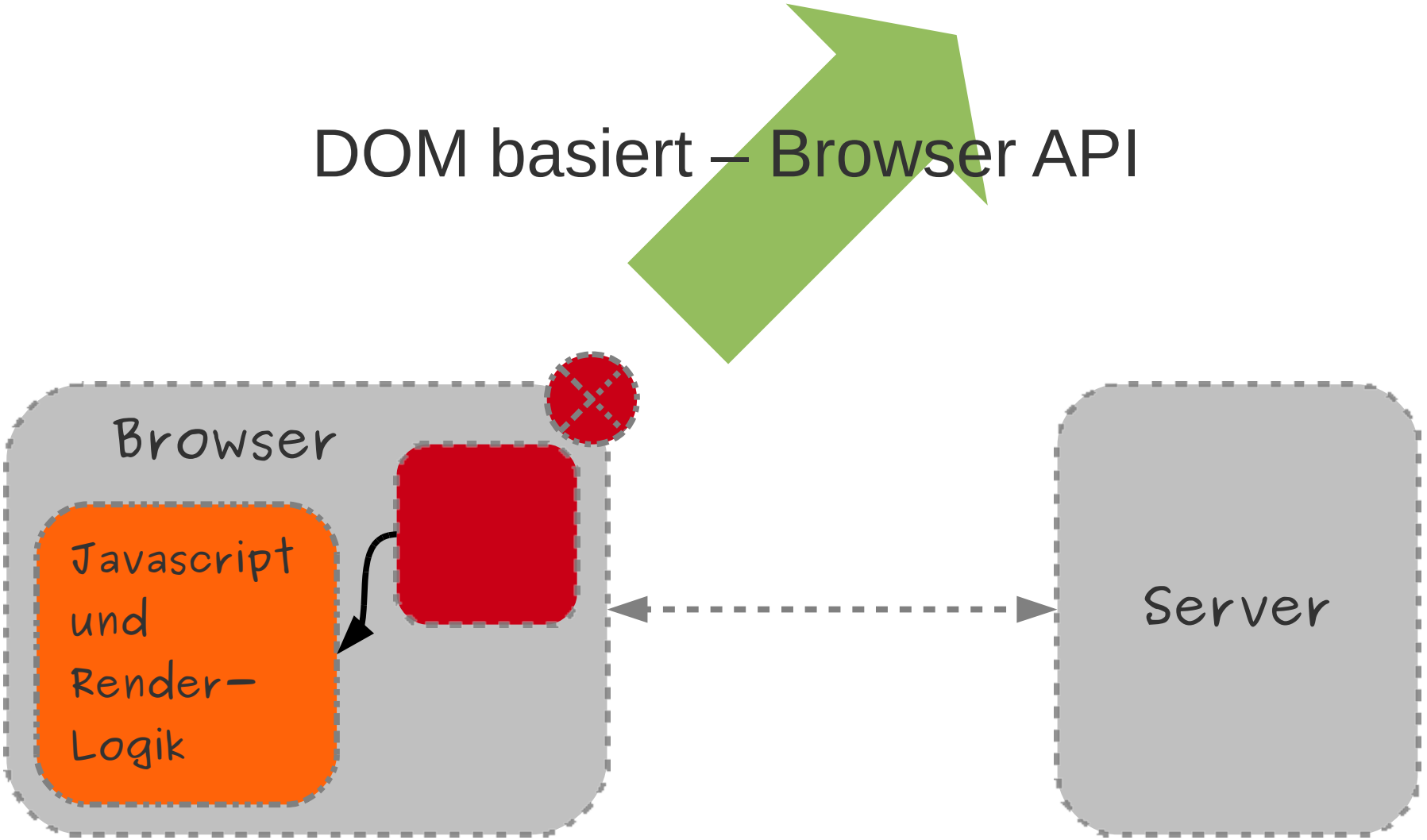
SeleniumRC

DOM basiert – Proxy

- Invasiver Testansatz
- Eingeschränkter Zugriff auf die Anwendung

Klassifizierung Testmethoden

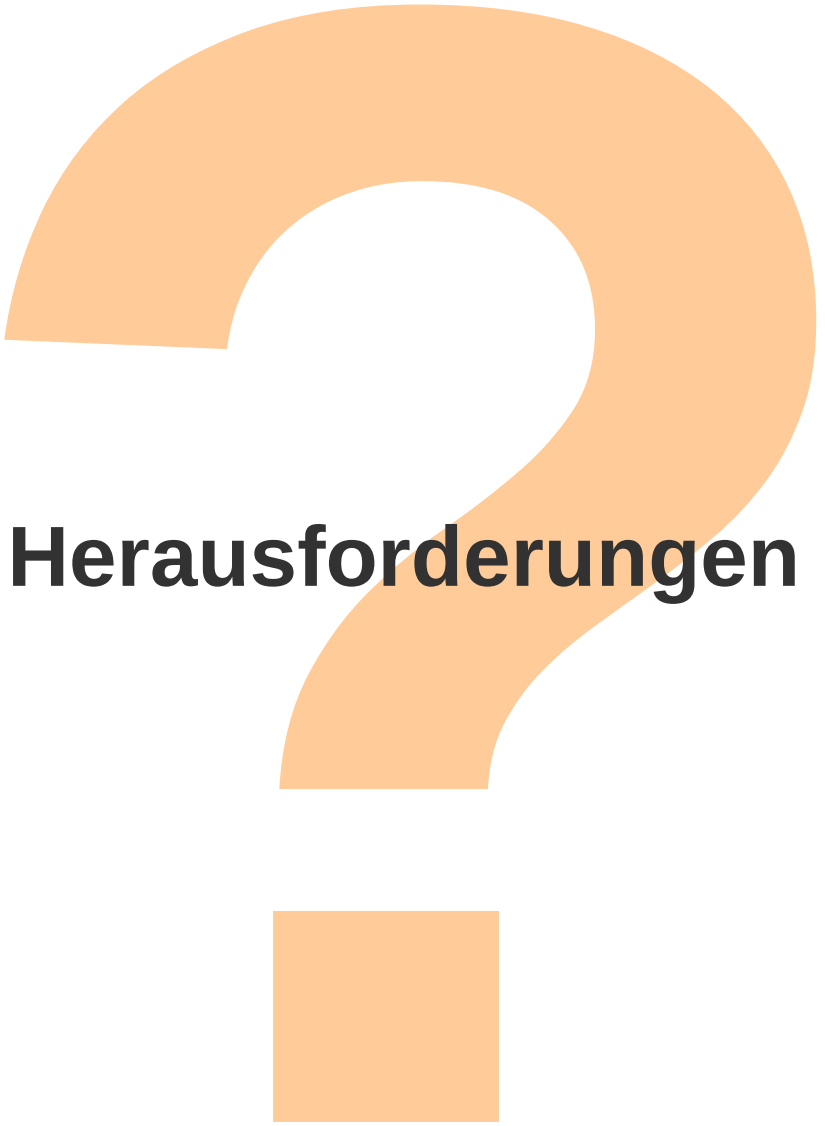
DOM basiert – Browser API



Selenium2.0, Molybdenum

DOM basiert – Browser API

- Nicht verfügbar für alle gängigen Browser
- Aufwändige Herstellung, da browserabhängig



Welche Herausforderungen gibt es?

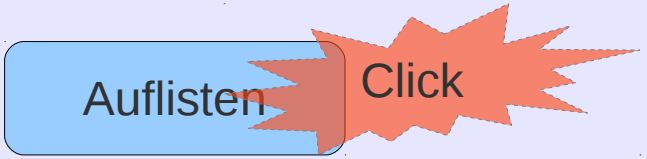
Zu lösende Probleme

- Synchronisation mit eventbasierten Abläufen
- Unterstützung für Akzeptanztests durch den Endnutzer
- Tests und Reports für den Endnutzer verständlich
- Testsetup für Akzeptanztests
- Alternative Medien unterstützen

selab.fbk.eu/tonella/papers/wqvv2007.pdf

Synchronisation mit Events - eine technische Lösung

Traditionelles Verfahren (I)



Element1	10
Element2	9
Element3	4
Element4	12

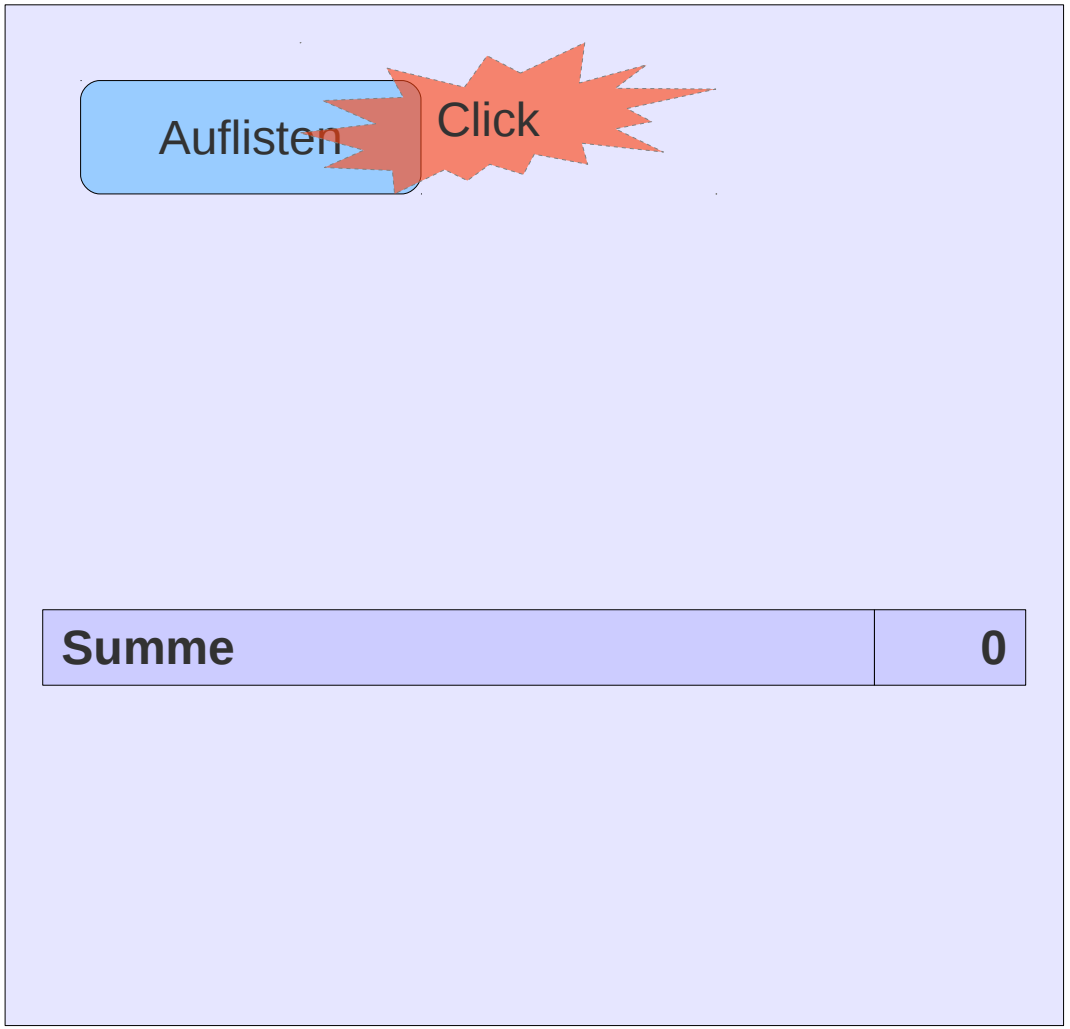
Summe	**35**

click 'Auflisten'

waitForTextPresent 'Summe'

- [x] assertText … '35'

Traditionelles Verfahren (II)



click 'Auflisten'

waitForTextPresent 'Summe'

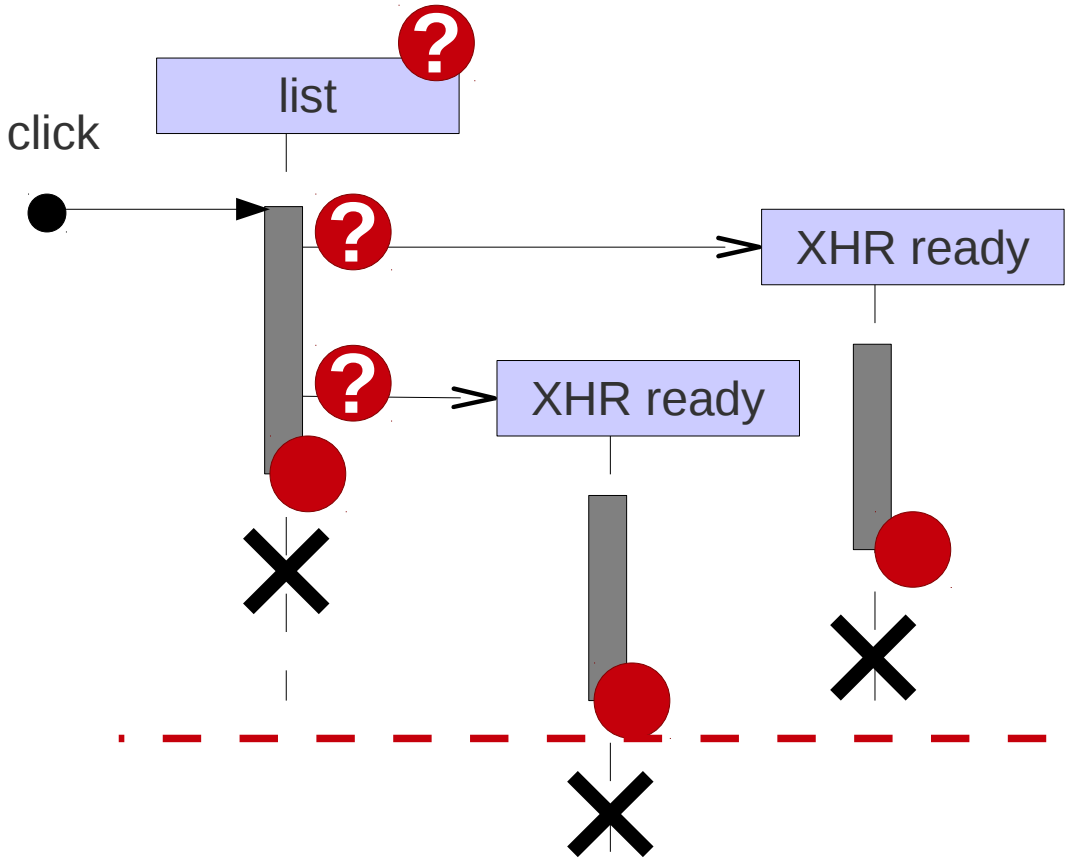
assertText … '35'

Traditionelles Verfahren - Fazit

Zur Synchronisation wird auf das Erscheinen eines Oberflächenelements gewartet.

Verfahren ist nicht stabil. Zur Stabilisierung sind Kenntnisse über die Implementierung notwendig.

Alternatives Verfahren



Auflisten

Element1	10
Element2	9
Element3	4
Element4	12
Summe	**35**

ajaxClick 'Auflisten'

assertText ... '35'

Demo

Synchronisation mit Events und XHR

Demo

Molybdenum Reports und ReRun