canoo

[delivering end-user happiness]

Mit dem Page Object Pattern und Geb zu Test Driven Development für Web-UIs

Stephan Classen

Über mich

Stephan Classen

- TDD enthusiast
- Liebt Open Source
- Hasst repetitive Aufgaben

Über mich

Stephan Classen

- TDD enthusiast
- Liebt Open Source
- Hasst repetitive Aufgaben
- Ein bisschen paranoid

Über mich

Stephan Classen

stephan.classen@canoo.com

Aktuelle Folien

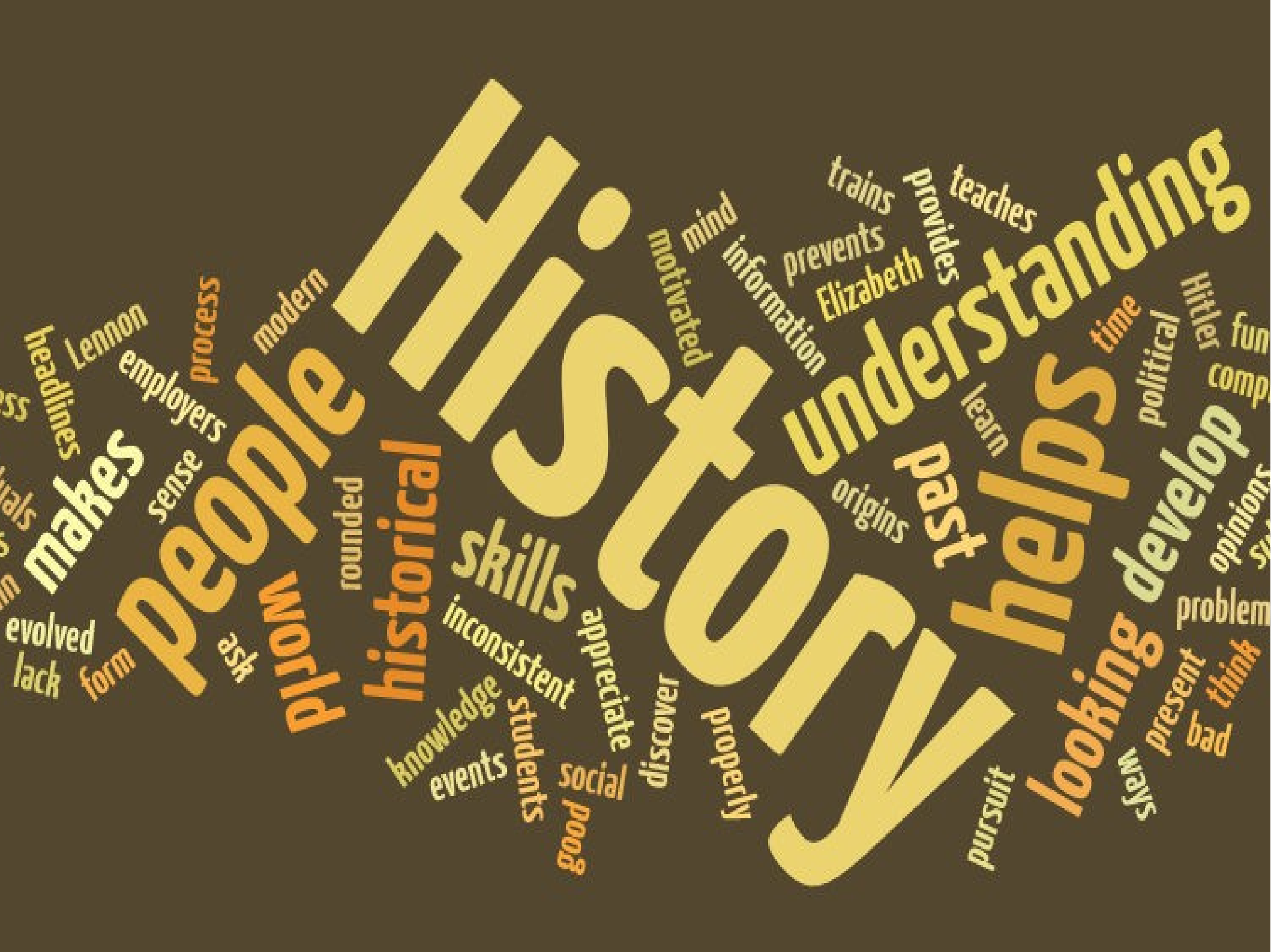
github.com/sclassen/herbstcampus2016

Vorteile von UI Tests

- End To End – Kein Mocking
- Erkennt Probleme einzelner Browser
- Schnellere Iterationen
- Spart Geld

Herausvorderungen bei UI Tests

- Schwierig zu automatisieren
- Stabilität
- Anfällig für Änderungen
- Langsam


History
people
understanding
helps
looking
develop
makes
world
skills
historical
past
trains
teaches
provides
prevents
Elizabeth
mind
motivated
information
Lennon
headlines
employers
process
modern
sense
rounded
ask
form
evolved
lack
inconsistent
knowledge
events
students
social
good
discover
properly
appreciate
origins
learn
time
political
Hitler
fun
opinions
problem
think
bad
present
ways
pursuit

Aufzeichnen - Abspielen

Beispiel: Selenium IDE

- Einfach zu erzeugen
- Schwierig nachzuvollziehen
- Schwierig anzupassen
- Schwierig zu erweitern

Skripten

Beispiel: Canoo WebTest / Selenium

- Nachvollziehbar und anpassbar
- Sehr Low-Level
- Viel Duplizierung wenig Reuse
- Tests kennen die UI Repräsentation

Modell basiert

Beispiel: WebDriver mit PageObjects

- Weniger Low-Level
- Abstraktion der UI Repräsentation
- Einfacher anpassbar

Natürliche Sprache

Beispiel: Cucumber / JBehave

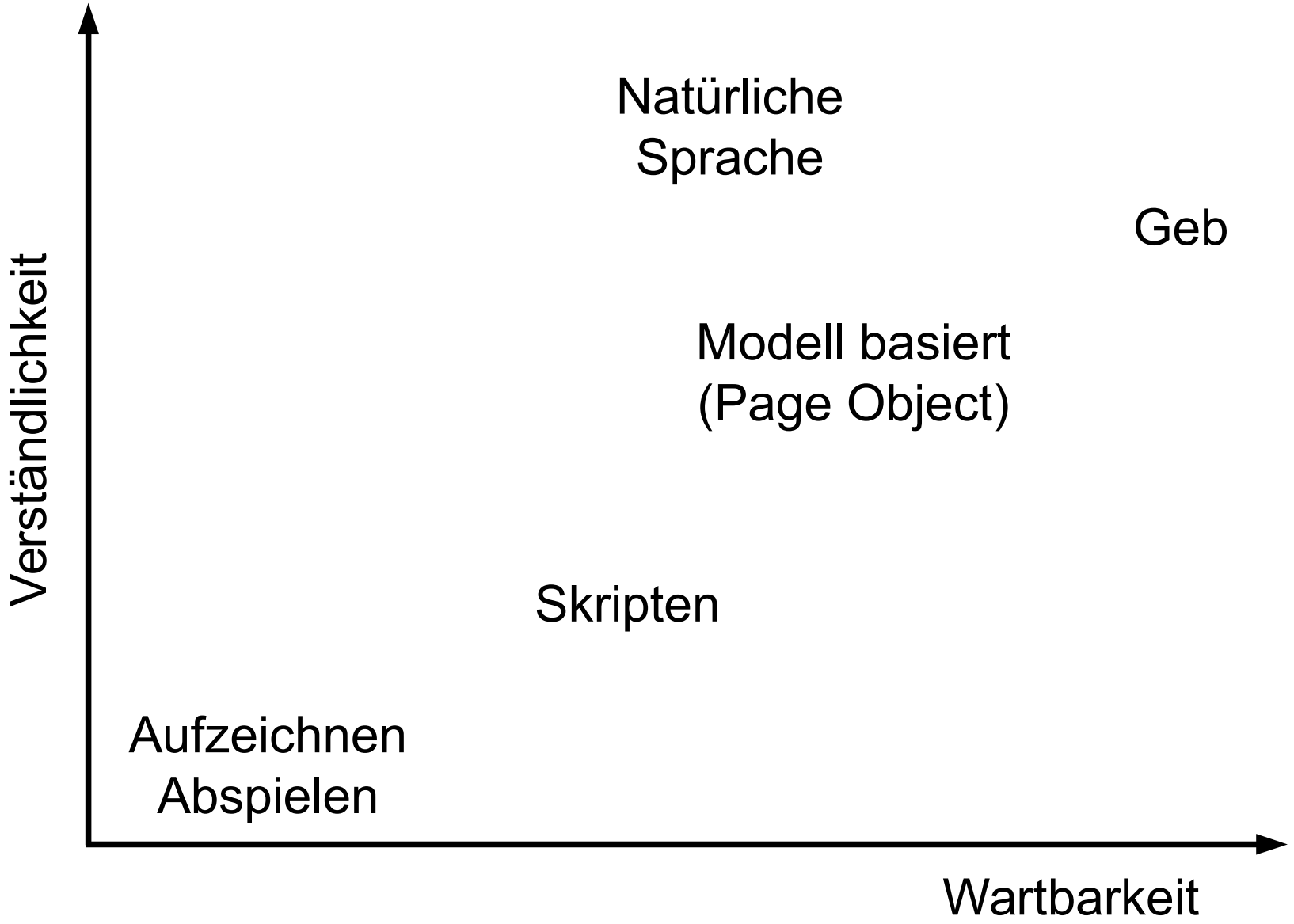
- Schicht über Skripting/Modell basiert
- Tests sind für alle verständlich
- Kosten für Übersetzung

Entwickler fokusiert

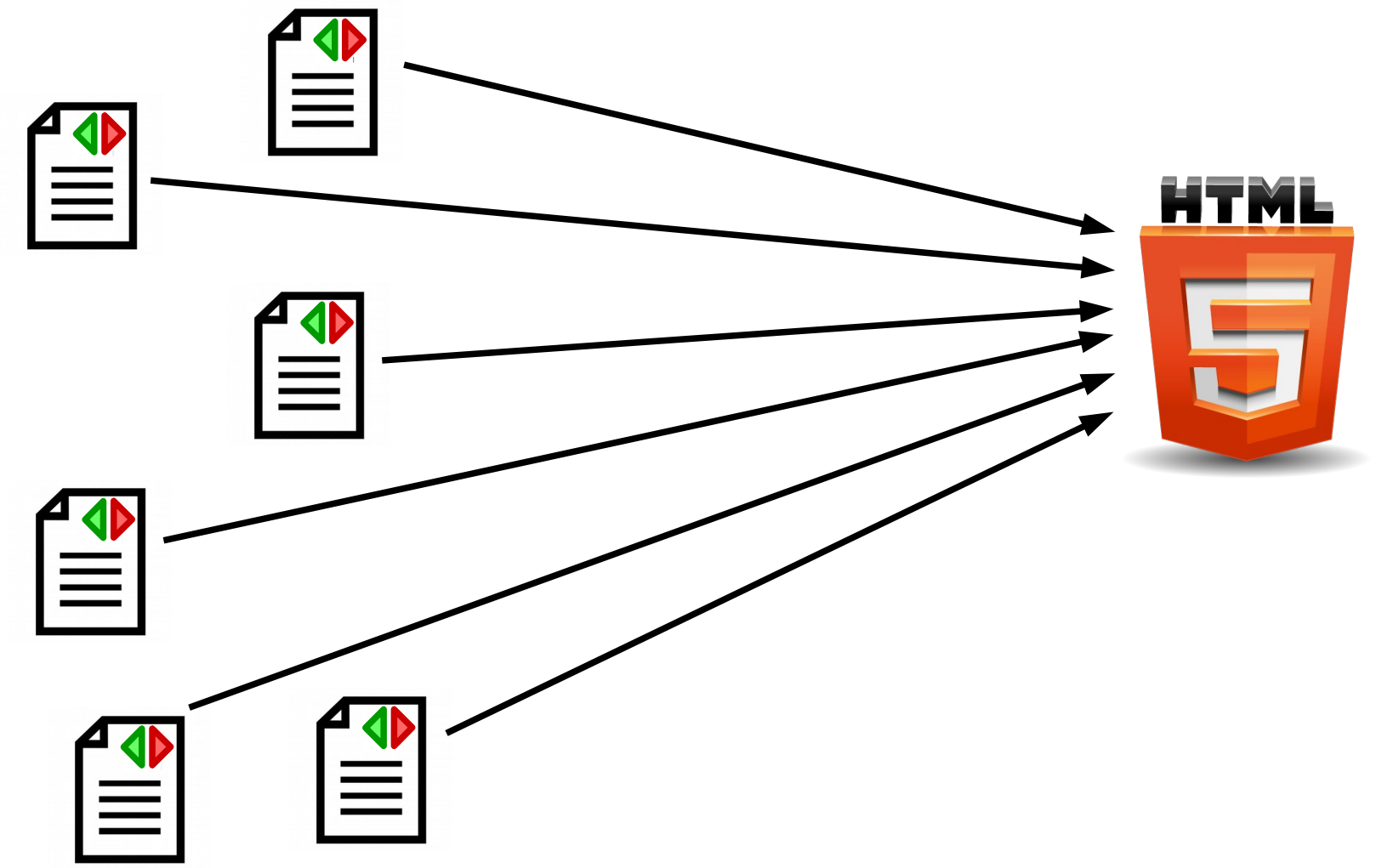
Beispiel: Geb (mit Spock)

- Groovy - Code mit starken DSLs
- Integrierte Modellunterstützung
- Lesbar für nicht IT Personen
- Fokus auf Bedienbarkeit
  - Jquery Selektoren
  - Detailierte Fehlermeldungen

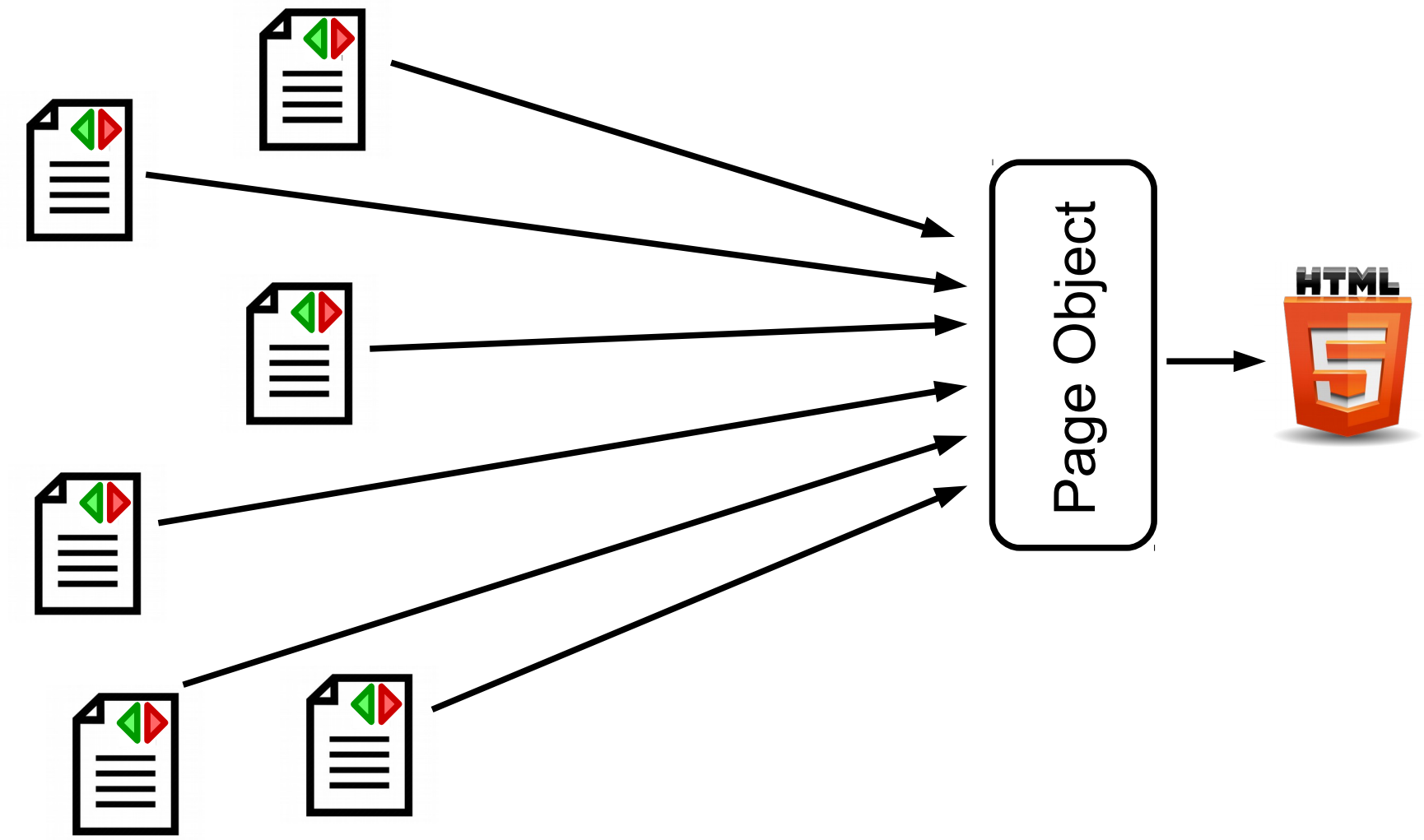
Ansätze in der Übersicht

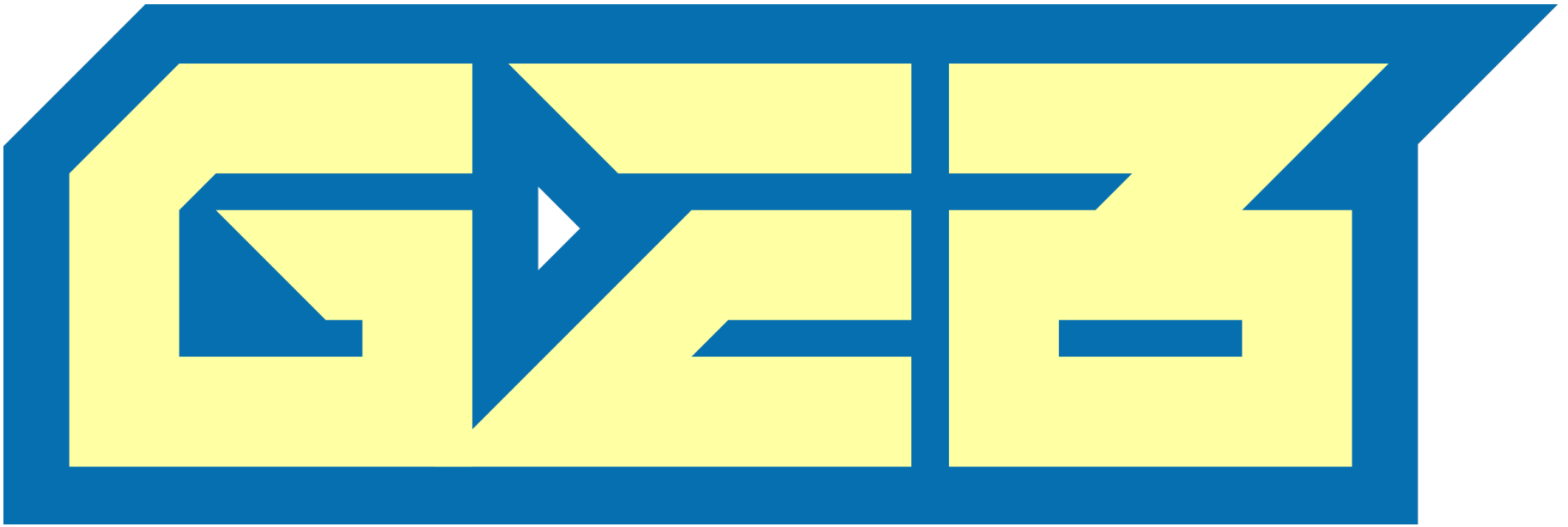


Ohne Page Object Pattern



Mit Page Object Pattern

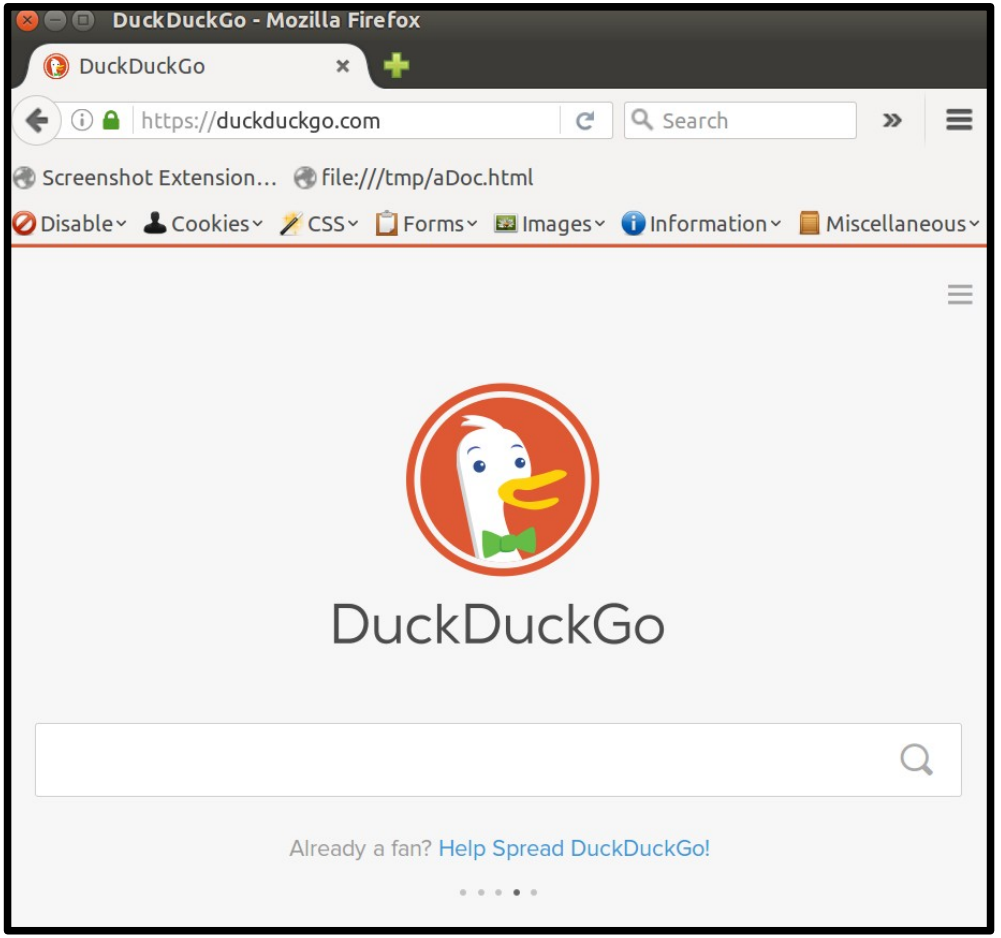




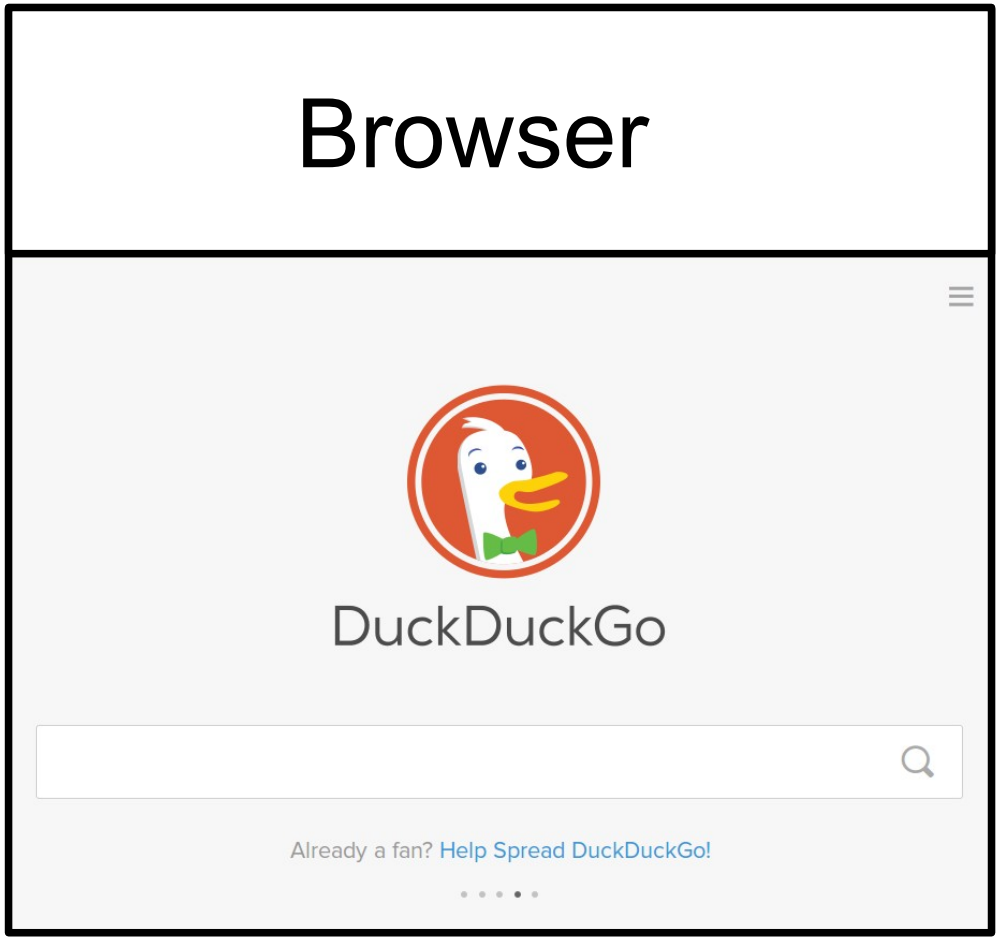
Geb – Daten und Fakten

- Sprache: Groovy
- Anfang: November 2009
- Gründer: Luke Daley
- Verantwortlicher: Marcin Erdman

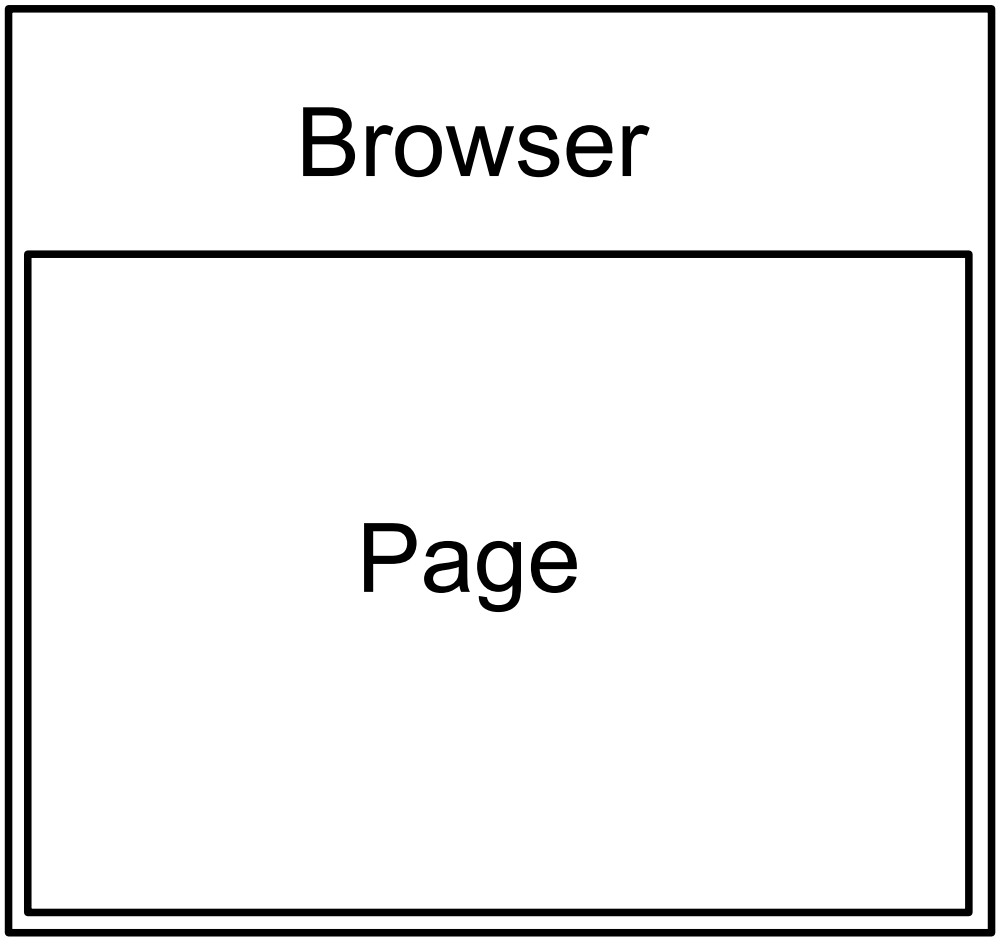
Geb – Architektur



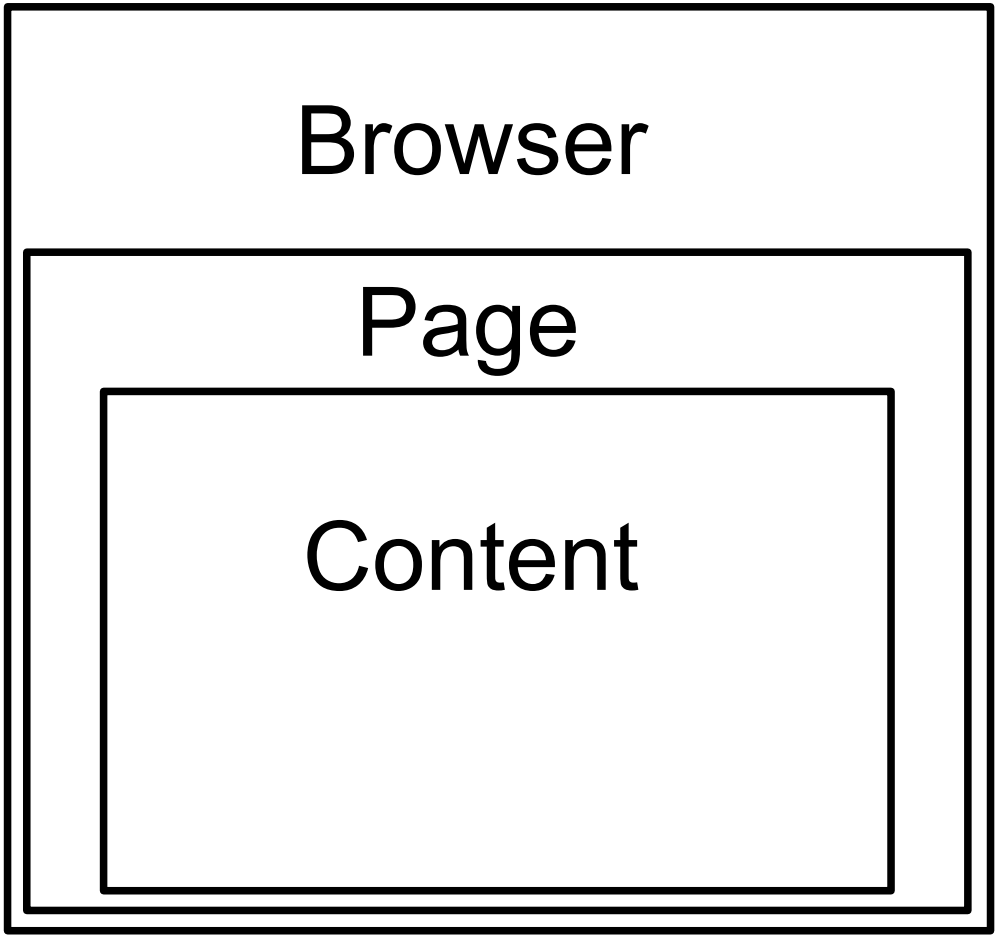
Geb – Architektur

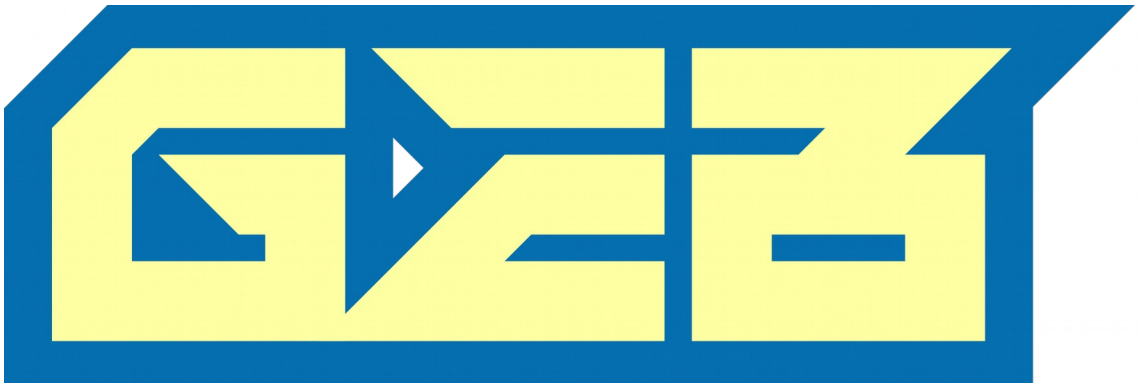


Geb – Architektur



Geb – Architektur





Content

Geb – Navigator API

Ein Navigator Objekt repräsentiert
0, 1 oder mehrere HTML Elemente.

- » Selektieren
- » Traversieren
- » Auslesen
- » Interagieren

Geb – Navigator API – Selektieren

```
$(<css>, <index/range>, <attribute/text>)
```

- Ermöglicht Elemente zu selektieren
- Alle Parameter sind optional
- Gibt nie NULL zurück

Geb – Navigator API – Selektieren

```
$(<css>, <index/range>, <attribute/text>)
```

» `$('div')`
```
// alle <div> Elemente
```

» `$('div.foo')`
```
// alle <div class="foo"> Elemente
```

» `$('div#foo p:first-child[title^="bar"]')`
```
// Falls vom WebDriver unterstüzt
```

Geb – Navigator API – Selektieren

```
$(<css>, <index/range>, <attribute/text>)
```

» `$('div', 0)`
`// das erste <div> Elemente`

» `$('div', 2..4)`
`// das 3., 4. und 5. <div> Elemente`

Geb – Navigator API – Selektieren

```
$(<css>, <index/range>, <attribute/text>)
```

» `$('div', attr: 'foo')`
```
// alle <div attr="foo"> Elemente
```

» `$('div', text: 'foo')`
```
// alle <div>foo</div>
```

» `$('div', attr: 'foo', text: 'bar')`
```
// alle <div attr="foo">bar<div>
```

Geb – Navigator API – Selektieren

```
$(<css>, <index/range>, <attribute/text>)
```

» `$('div', text: ~/.*foo/)`
  `// Regexp auf attribute/text`

» `$('div', text: contains('foo'))`
  `// Hilfsmethode erzeugt eine Regexp`

Geb – Navigator API – Selektieren

```
» $('div#foo').find('form')
  // Sucht innerhalb des Navigator
  // Gleiche Parameter wie $()

» $('div').filter('.b')
» $('div').not('.b')
» $('div').has('p')
» $('div').hasNot('p', attr: 'foo')
  // Selektiert eine Teilmenge des
  // Navigator
```

Geb – Navigator API – Traversieren

- `$('#foo').parent()`
- `$('#foo').children('div', text: 'bar')`

```
// Nicht rekursiv
```

- `$('#foo').previous()`
- `$('#foo').prevAll()`
- `$('#foo').next()`
- `$('#foo').nextAll()`
- `$('#foo').siblings()`

```
// Geschwister / Nachbarn
// Unterstützen <css> und <attribute>
```

Geb – Navigator API – Auslesen

```
<p id="a" class="b c" foo="d">Inhalt</p>
```

```
» $('#a').text()          == 'Inhalt'
» $('#a').tag()           == 'p'
» $('#a').classes()       == ['b', 'c']
» $('#a').attr('foo')     == 'd'
» $('#a').@foo()          == 'd'

» $('input').value()
  // Inhalt eines Formular Element
```

Geb – Navigator API – Auslesen

```
» $('#foo').isDisplayed()
  // ist das Element sichtbar

» $('input#bar').isEnabled()
» $('input#bar').isDisbled()
  // Zustand des @disabled Attribut

» $('input#bar').isEditable()
» $('input#bar').isReadOnly()
» // Zustand des @readonly Attribut
```

Geb – Navigator API – Interagieren

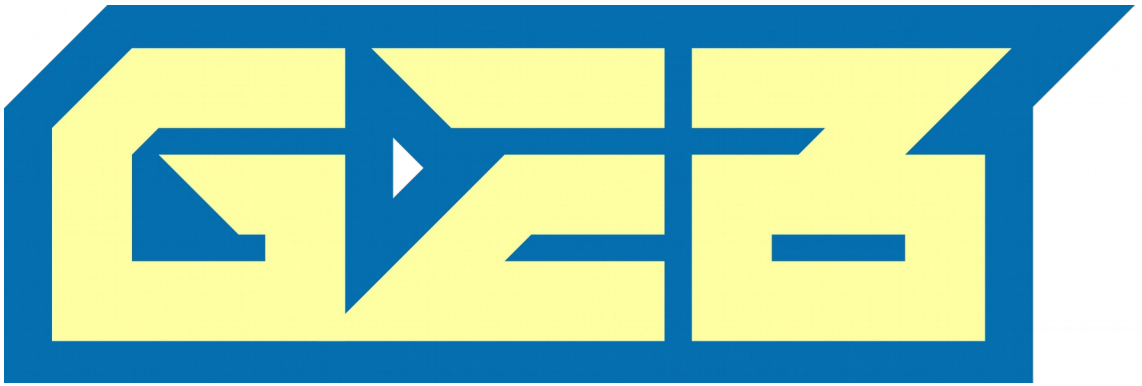
- `$('#foo').click()`
  `// einfach immer drauf klicken`

Geb – Navigator API – Interagieren

- `$('input#username').value('test')`
- `$('input#username') = 'test'`
  `// Schreiben (Ersetzen)`

- `$('input#username') << 'user'`
  `// Schreiben (Anfügen)`

- `$('input#pwd') <<`
  `Keys.chord(Keys.CONTROL, 'v')`
  `// Auch Tastenkombinationen möglich`



Page

Geb – Page

```
class MyPage extends Page {
    static url = '/foo'          (1)
    static at = {
        title == 'My Page'       (2)
    }
}
```

1. URL der Seite
2. Überprüft ob der Browser auf dieser Seite ist

Geb – Page – Content

```
class MyPage extends Page {
    static content = {
        header { $('h1', 0) }
        footer { $('div#footer') }
    }
}
```

» Mit dem Content Block wird der Inhalt der Seite modelliert.

Geb – Page – Content

```
class MyPage extends Page {
    static content = {
        itemCount { $('.item').size() }
        name { $('input#name').value() }
    }
}
```

» Ist meistens ein Navigator Objekt muss aber nicht !!

Geb – Page – Content

```
class MyPage extends Page {
    static content = {
        foo { arg ->
                $('div', title: arg) }
    }
}
```

» Es können auch Argumente übergeben werden

Geb – Page – Content

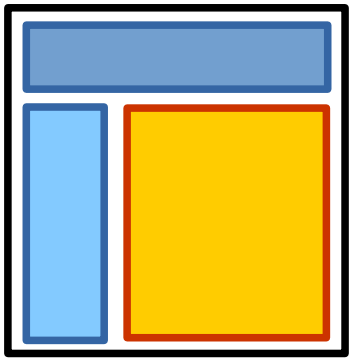
```
class MyPage extends Page {
    static content = { … }

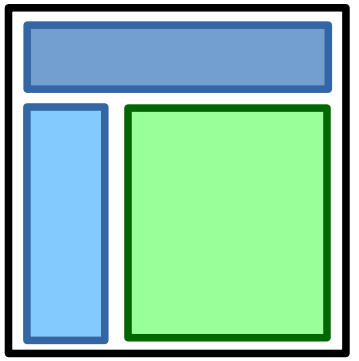
    void foo() {
        // kann content verwenden
    }
}
```

» Neben content können auch normale Methoden definiert werden.

Geb – Page – Content

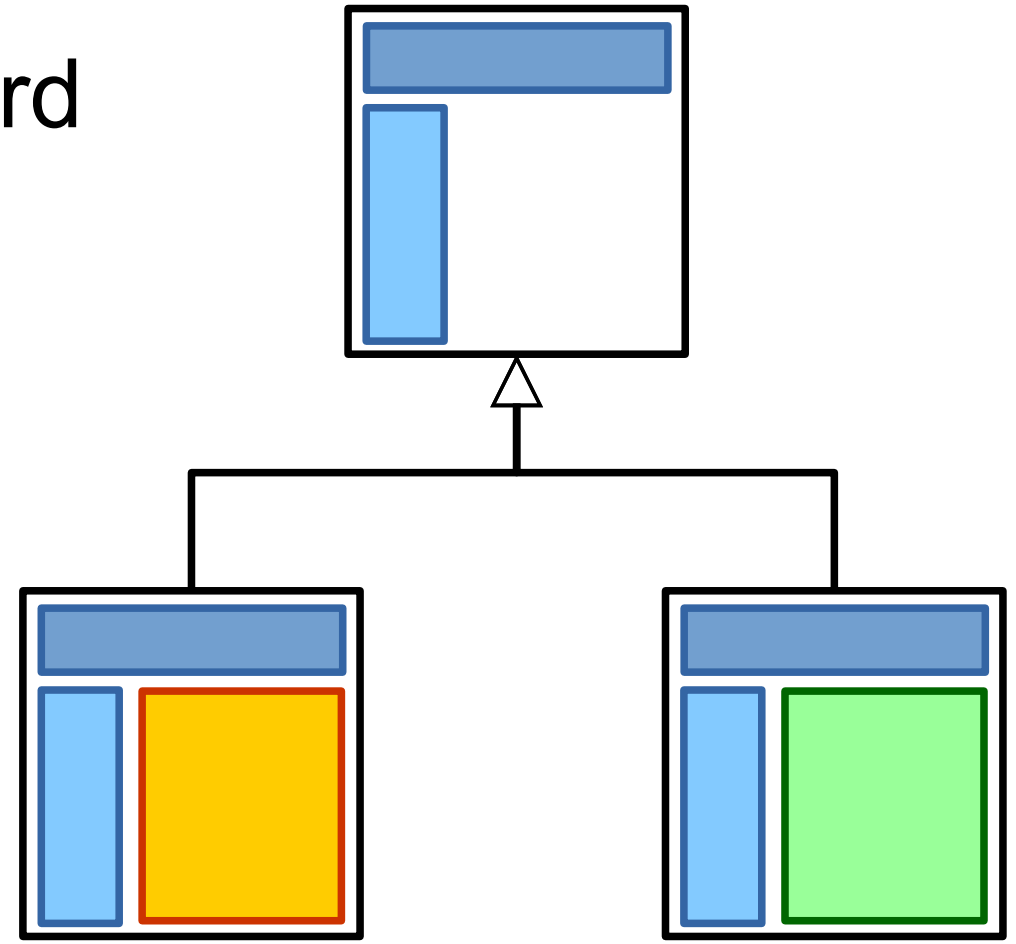
» Der Content wird vererbt.



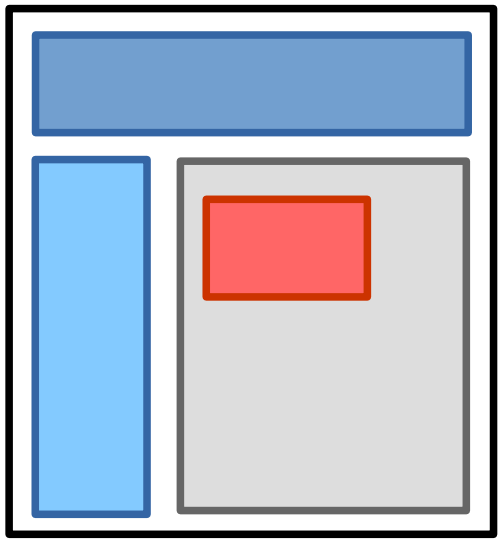


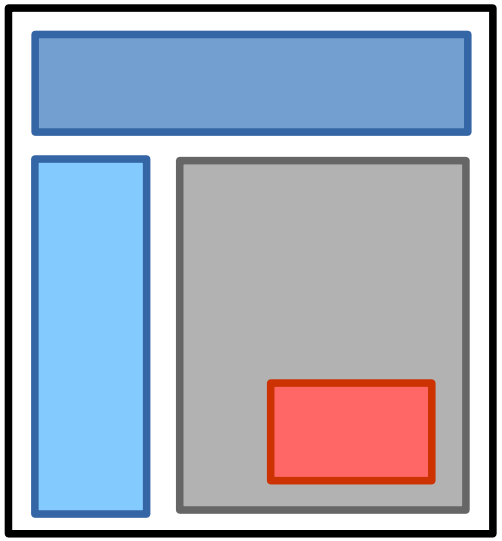
Geb – Page – Content

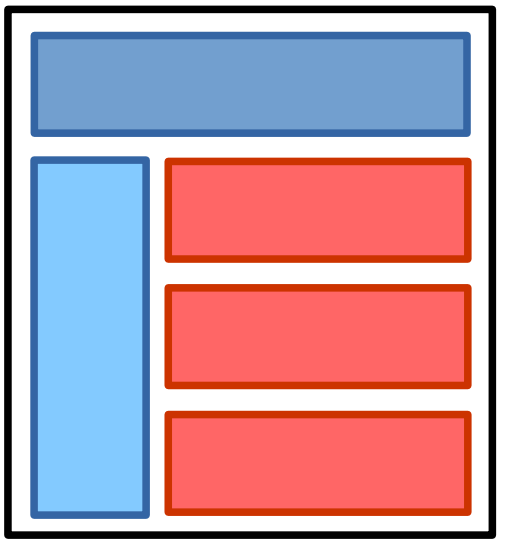
» Der Content wird vererbt.



Geb – Module







Geb – Module

```
class MyModule extends Module {
    static content = {
        name { $('td', 0) }
        price { $('td', 1) }
    }
}
```

» Gruppiert mehrere Elemente

» Wie eine Page aber ohne URL

Geb – Module

```
class MyPage extends Page {
    static content {

        shoppingCart {
            $('div#cart').module(Cart)
        }

        searchHits {
            $('tr').moduleList(SearchHit)
        }
    }
}
```

Geb – Page – Helfer

```
» browser.driver.executeScript(
      "return arguments[0];", 1
  ) == 1
```

```
» js.aGlobalVariable == 'foo'
» js."document.title" == 'about'
```

```
» js."document.write"('<div>foo</div>')
```

```
» js.exec(1, "return arguments[0];") == 1
```

Geb – Page – Helfer

```
» interact {
      dragAndDropBy($('#foo'), 150, 200)
  }
```

```
» interact {
      keyDown Keys.SHIFT
      click $('li.clicky')
      keyUp Keys.SHIFT
  }
```

Spock – Daten und Fakten

- Sprache: Groovy
- Version 0.1: März 2009
- Gründer: Peter Niederwieser

Spock

```
class MySpec extends Specification {

    def "test something"() {
        given:
          int a = 2

        when:
          int result = 2 * a

        then:
          result == 4
    }
}
```

Spock

```
class MySpec extends Specification {

    void setup() { … }       // @Before
    void setupSpec() { … } // @BeforeClass

    void cleanup() { … }       // @After
    void cleanupSpec() { … } // @AfterClass

}
```

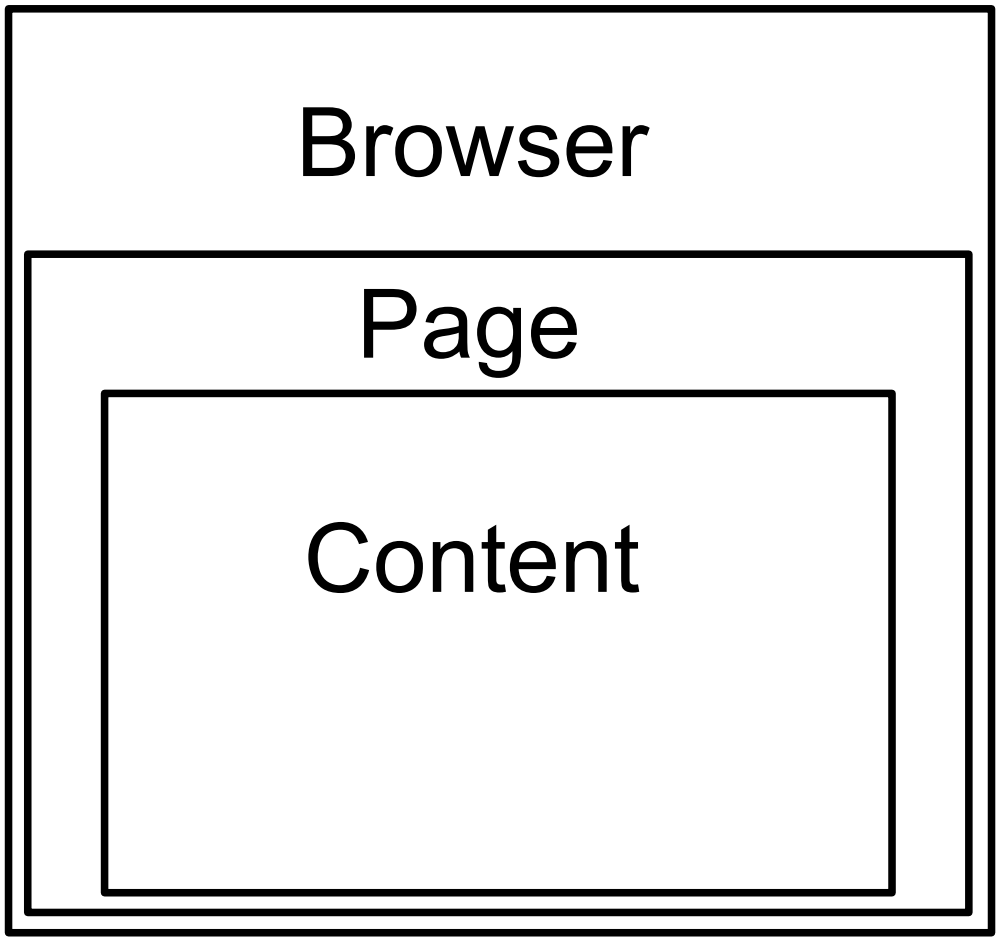
Spock & Geb

```
@Stepwise
class MySpec extends GebReportingSpec {

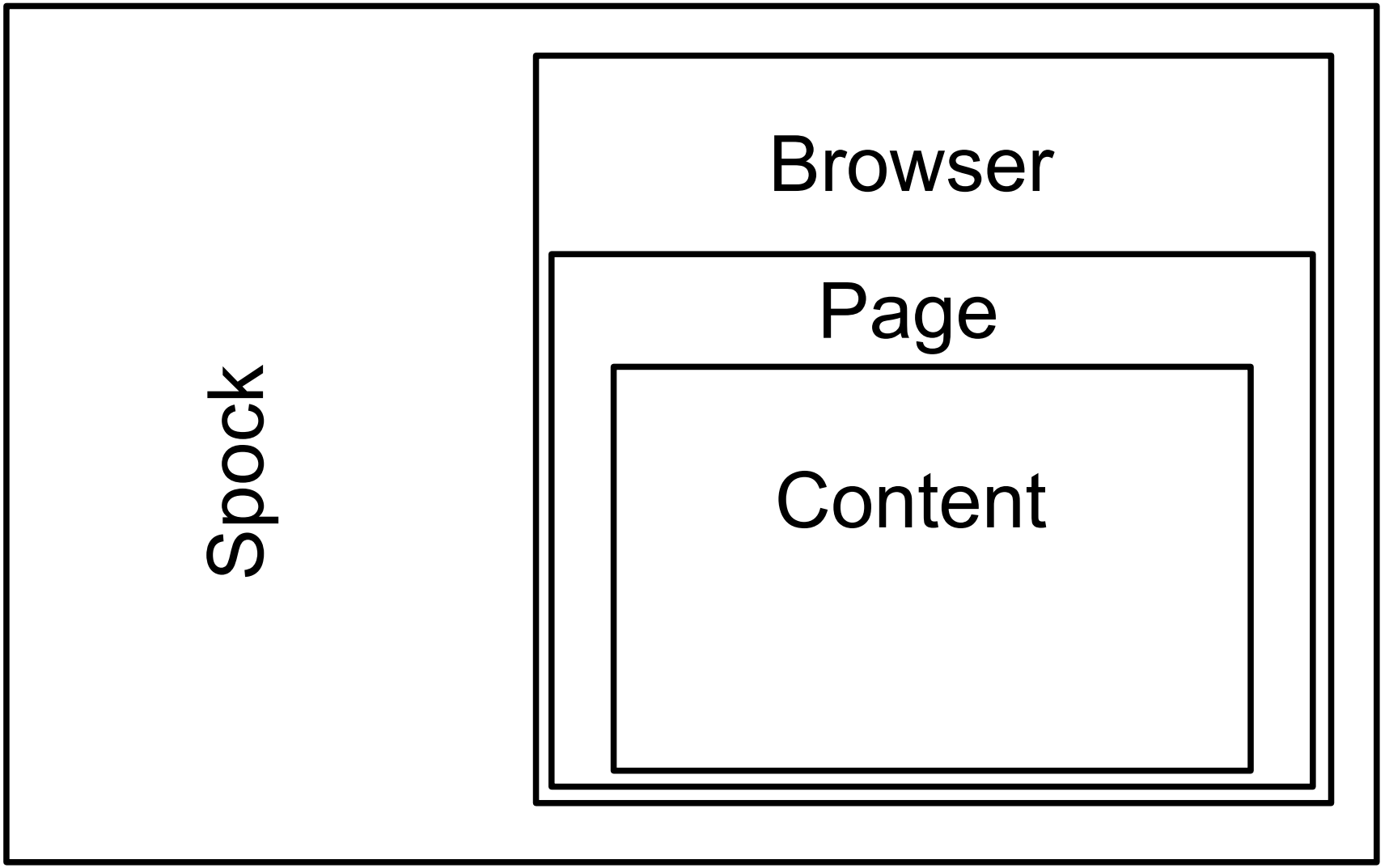
    def "smoke test"() {
        when:
          go 'localhost:8080/foo'

        then:
          title == 'Hello World'
    }
}
```

GEB & Spock



GEB & Spock



Aus dem Leben eines Hotline-Mitarbeiters

Single Page Application

Synchron vs. Asynchron

- Bei klassischem Laden einer Seite weiss der Browser wann die Seite geladen ist.

- Bei asynchronem (Nach-)Laden von Daten/Elementen weiss der Browser nicht wann die Seite geladen ist.

Warten warten warten

- Warten bis die Seite „geladen“ ist
- Die Geb Page bietet Hilfestellung

```
$('a#loadMore').click()

waitFor {
    items.size() > 10
}

items[10].click()
```

Worauf warten?

- Block UI
  - Elemente werden hinter dem Overlay geladen

- Alle Request fertig
  - Counter im JS verwenden

Login schlägt fehl

KEEP
CALM
AND
LOGOUT AND
LOG BACK IN

Login schlägt fehl

- Am Ende jedes Tests ausloggen
  - In der cleanupSpec Methode
  - In einer Basisklasse

Daten sind alle weg

Daten sind alle weg

- Erzeuge die Daten für den Test
  - In der setupSpec Methode
  - Räume in der cleanupSpech auf
  - Verwende unique Strings (Zeit/Hash)
  - Builder Pattern

Works on my machine

Works on my machine

- Laufzeitinfo in der Fusszeile anzeigen
  - Server / DB Name
  - Version (inkl. Commit-ID)


geek & poke
PLEASE DON'T ASK ME HOW THIS WORKS AND PLEASE DON'T ASK ME WHY THIS WORKS!!

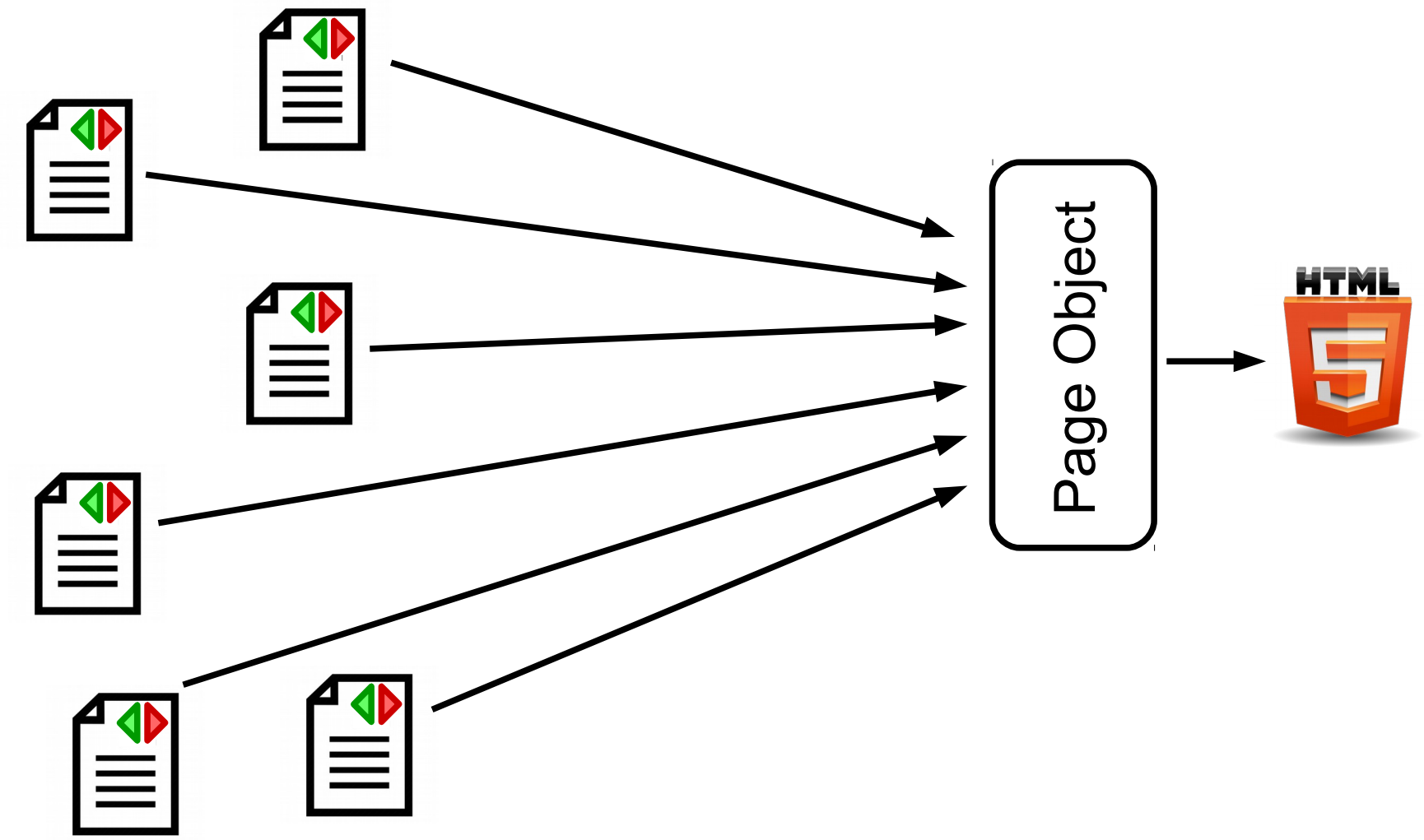
HTML markieren

- Id und Klassen verwenden um wichtige Elemente zu markieren.
- Aussagekräftige Namen verwenden

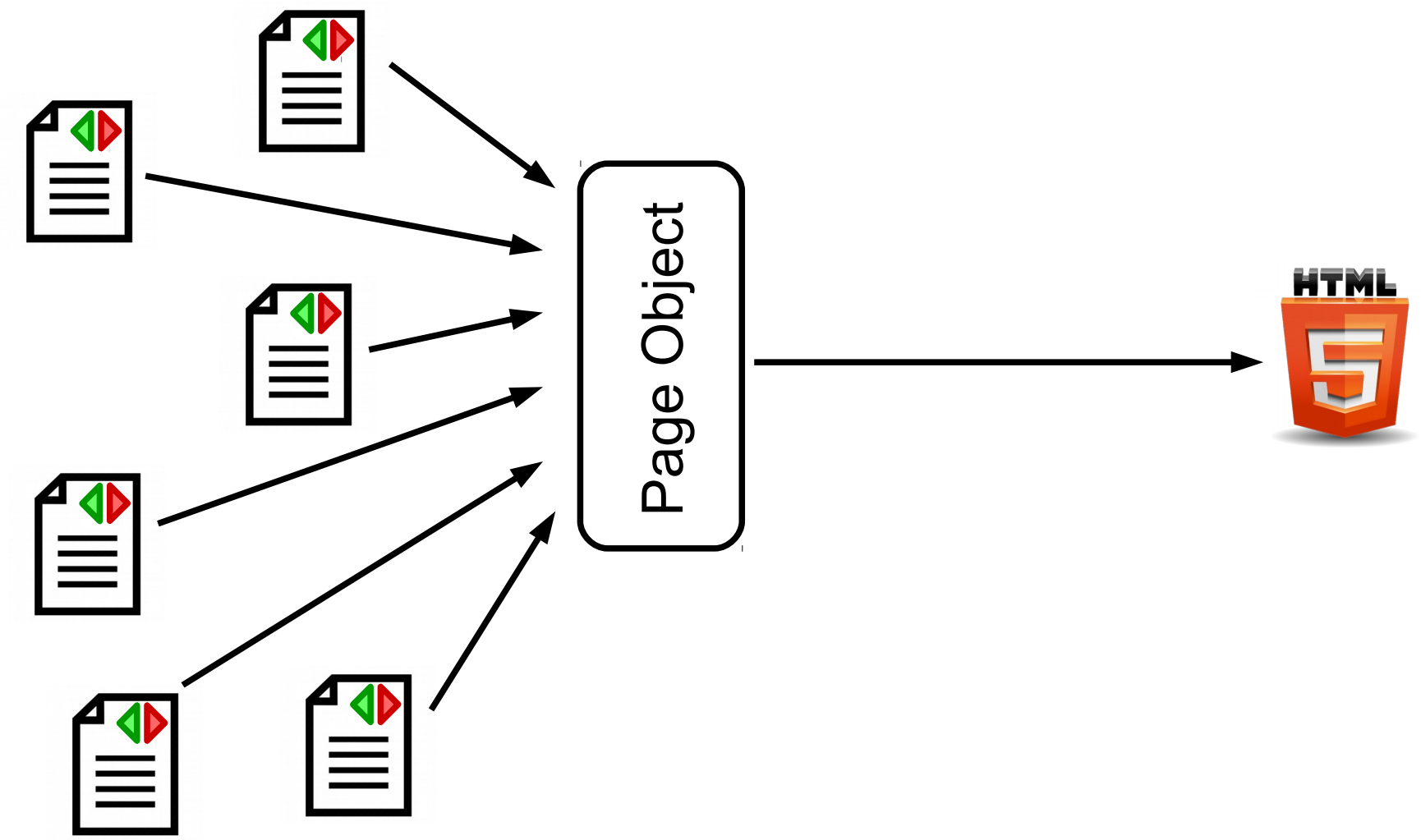
Strukturiert Code

- Verwendet Basisklassen für Page- und für Testklassen
- Geb-Module für Listen und Pop-Ups

Wartbarkeit



Wartbarkeit



Page Object näher an den Test

- Content der Page soll dem Test die Information in der richtigen Form liefern
  - Meistens keine Navigatoren
  - Parametrieter Content

Page Object näher an den Test

- `titleLabel.text().contains(user.name)`
- `userName == user.name`

- `tableRows.hasNot('th').size() == 5`
- `numResults == 5`

- `tableRows[5].text().contains('foo')`
- `resultTitle(5) == 'foo'`

Page Object näher an den Test

- Methoden für Aktionen
  - Navigieren
  - Ausfüllen einfacher Formulare
  - Methode pro Ausgang/Resultat

Page Object näher an den Test

- ```
  aboutPageLink.click()
  waitFor { at AboutPage }
  ```
- ```
  gotoAboutPage()
  ```

- ```
  userNameInput = username
  passwordInput = password
  loginLink.click()
  waitFor { at MainPage }
  ```
- ```
  login(username, password)
  ```
- ```
  loginUnsuccessful(username, password)
  ```
  ```

Tool Support

- Geb und Spock sind Groovy
- Groovy ist eine dynamische Sprache
- IDE ist meist ein wenig überfordert

- Delegation
  - Spock → Browser → Page

Page explizit machen

```
@Shared
Page p

def "someTest"() {
    given:
      p = at MainPage

    when:
      p = p.gotoAboutPage()

    then:
      p.version == '1.2.6'
}
```

Schreibt UI-Tests mit Geb und Spock

- http://www.gebish.org
- http://spockframework.org