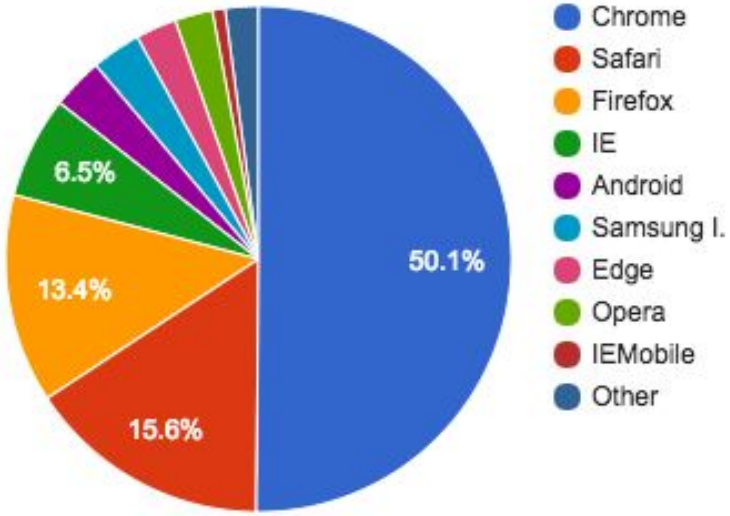
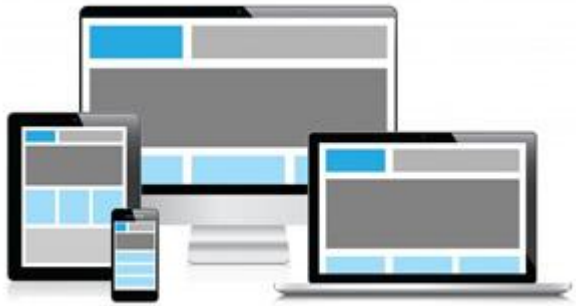
ThoughtWorks®

LAYOUT CHECKS WITH GALEN

Layout Checks



Browser usage in Europe within the last 3 months (desktop, tablet, mobile)



Layout Checks

- Manual testing
- Image comparison

- After development has finished
- Usually done/maintained by QA
- Static image vs dynamic content
- Hard to measure

Framework Requirements

- Visual testing
- Scalability
- Readability
- Error feedback
- Functional testing

What is Galen?

- Open source framework for automated layout checks
- Designed for testing responsive websites
- Checks object location in relation to other objects on page
- Based on Selenium
- Supports Java and JavaScript tests

Why Galen?

- Most tools for cross browser testing take screenshots or compare images
- Galen is based on design specifications
- Galen checks can be written before development even begins and describe the layout in human-readable language
- Elements can be described with different degrees of precision

How does Galen work?

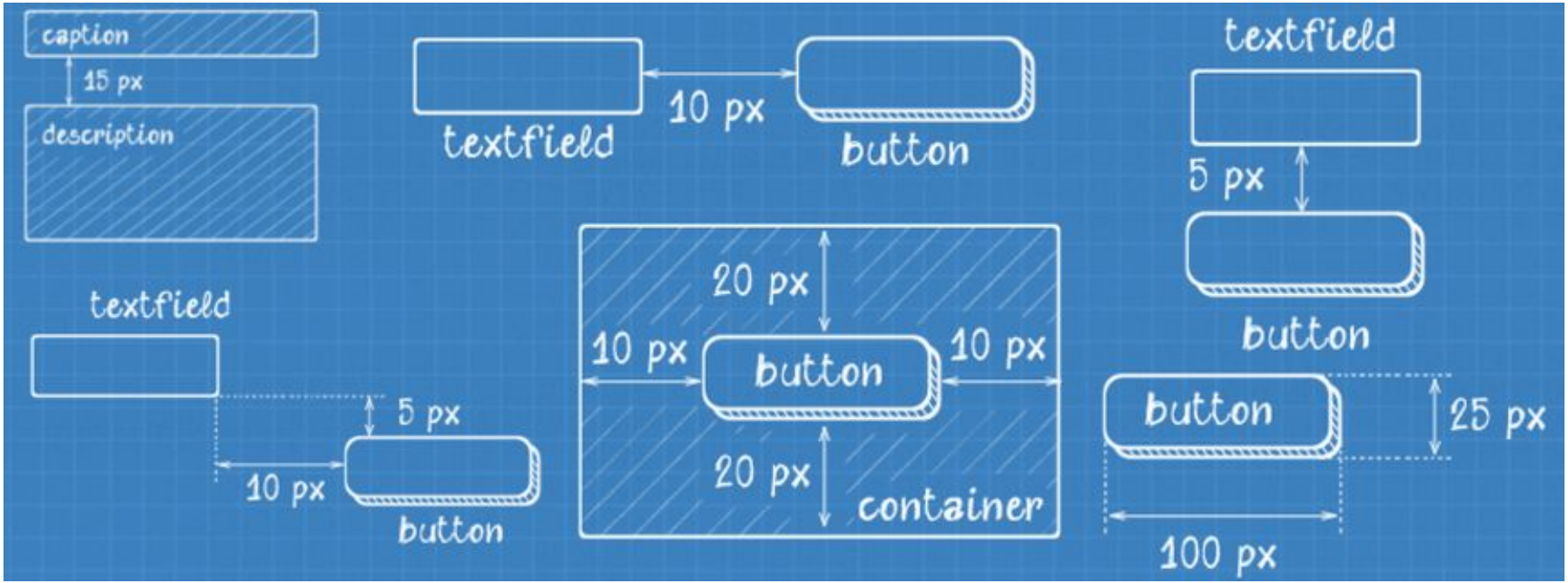
It works in the following way:

1. Galen opens a page in the browser
2. Resizes the browser to specified size
3. Tests the layout with Galen Specs

Galen uses Selenium for interacting with elements on page and getting their locations and dimensions.

Once Galen sees that something is wrong – it reports the error, takes a screenshot and highlights the misbehaving element on it.

How does Galen work?



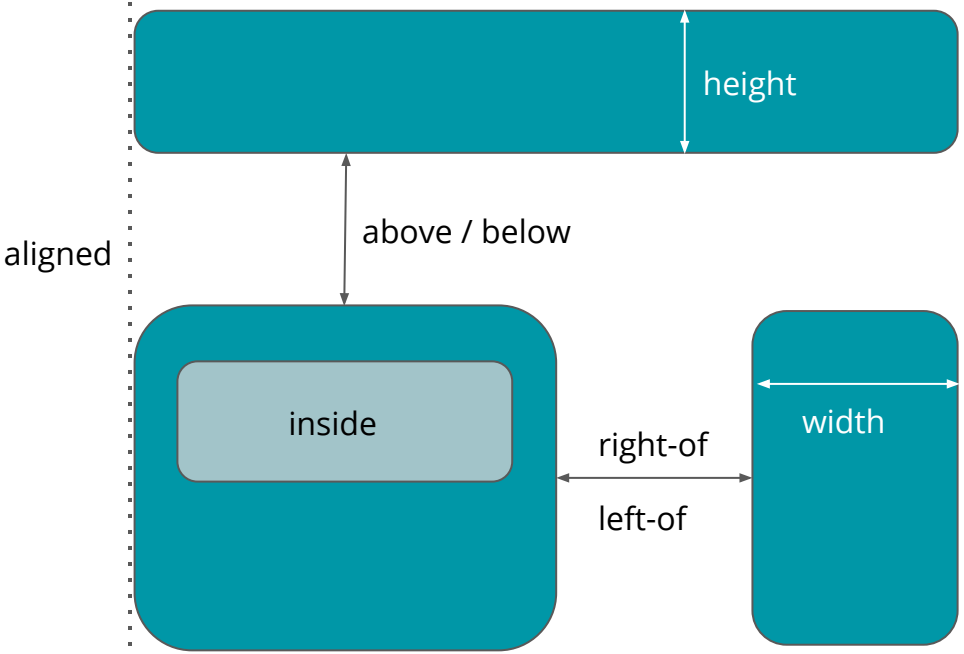
Galen syntax

Position
Above, below, right-of, left-of, inside, centered, aligned

Dimensions
Width, height

Visibility
Visible, absent

...



What are Galen Specs?

Syntax used to describe what page should look like for different devices

```
@objects
    headline            h1
    social-nav          div.social-sharing
    social-nav-item-*   ul.shareaholic-share-buttons li



= Insights content header section =

    headline:
        text is "SOFTWARE TESTING"
        below priority-nav
        above social-nav

    social-nav:
        above article-container
```

Galen Report

Galen Test Report

Tests | Groups

Test	Passed	Failed	Warnings	Total	Groups	Started	Duration	
Insights test on desktop emulation device	414	0	0	414		06-10-2016 14:51:21	18s	
Insights test on mobile emulation device	423	0	0	423		06-10-2016 14:50:15	22s	
Insights test on mobile emulation device	423	0	0	423		06-10-2016 14:50:38	22s	
Insights test on tablet emulation device	401	0	0	401		06-10-2016 14:51:00	20s	

Galen Report Details

Back to Test Overview | Expand All | Collapse All | Expand Errors

Insights test on desktop emulation device

14:51:21 INFO Open https://www.thoughtworks.com/insights/software-testing

14:51:40 - LAYOUT Check layout: specs/insights01.gspec included tags: desktop

Heat Map

- Header section
  - navbar should be above main-nav
  - main-nav should be above priority-nav and vertically aligned centered
    - **main-nav**
      - ✔ above priority-nav
      - ✔ aligned vertically centered priority-nav
  - main-nav-menu should be right of logo
  - mobile-nav should not be visible
  - main-nav-menu-item-* should be displayed next to each other and horizontally aligned
  - channels should be visible
- Insights content header section
- Insights Article section
- Article cards section

14:51:40 + INFO Click load more button to show more article

Galen Pages

- Javascript framework for page object pattern
- Easy way to add functional testing, e.g. click

Example

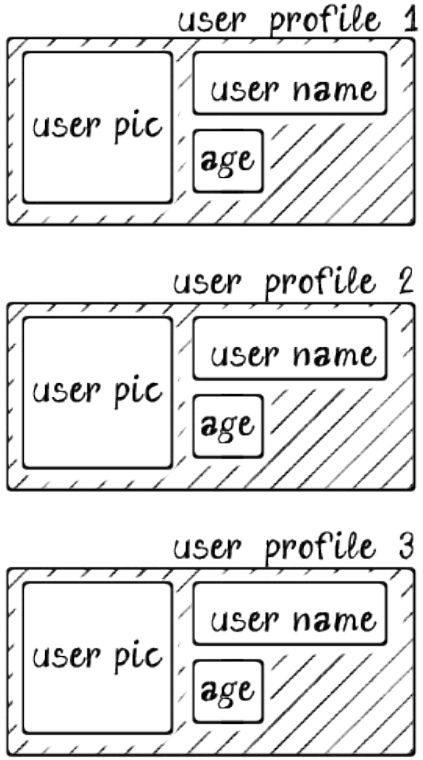
```
this.AccessibilityPage = $page("Accessibility page", {
     termsLink:   "xpath: //a[@href='/terms-and-conditions/']"
}, {
     goToTermsPage: loggedFunction ("Click terms link to open T&C page", function(){
          this.termsLink.click();
     })
});
```

```
accessibility = new AccessibilityPage(driver);
accessibility.goToTermsPage();
```

Components

- Reuse design specs
- Reduce maintenance costs
- Find design patterns and inconsistencies

```
user-profile-*:
    component component/user-profile.gspec
```



Custom rules

- Increases readability of spec
- Find design patterns and inconsistencies
- User-defined

Example

```
@rule %{objectPattern} should be displayed in 4 columns in %{container}
    @forEach [${objectPattern}] as item
        ${item}:
            width 22 to 25% of ${container}/width
```

```
= Article Header section =
    @on desktop
        | journal-name should be above article-section and vertically aligned left
        | article-cards should be displayed in 4 columns in main
```

DEMO

Why did we choose Galen?

Process

- helps to structure design & discover general layout patterns
- layout testing becomes more of a tool for test-driven development

Testing/Development

- Quickly run layout check in a multitude of browsers and sizes, mobile checks when set up with grid/appium
- Decouple layout checks from functional checks
- Reduce risk of side effect of layout changes

How does Galen fit into the process?

- Encourages communication and pairing between UX, QA and Dev
- Check layout as early as possible
- Separation of functional and visual checks
- Design documentation
- Local execution of layout checks by Dev and QA
- Included in CI pipeline: every push to dev triggers layout checks
- Specs were written by QA and Dev

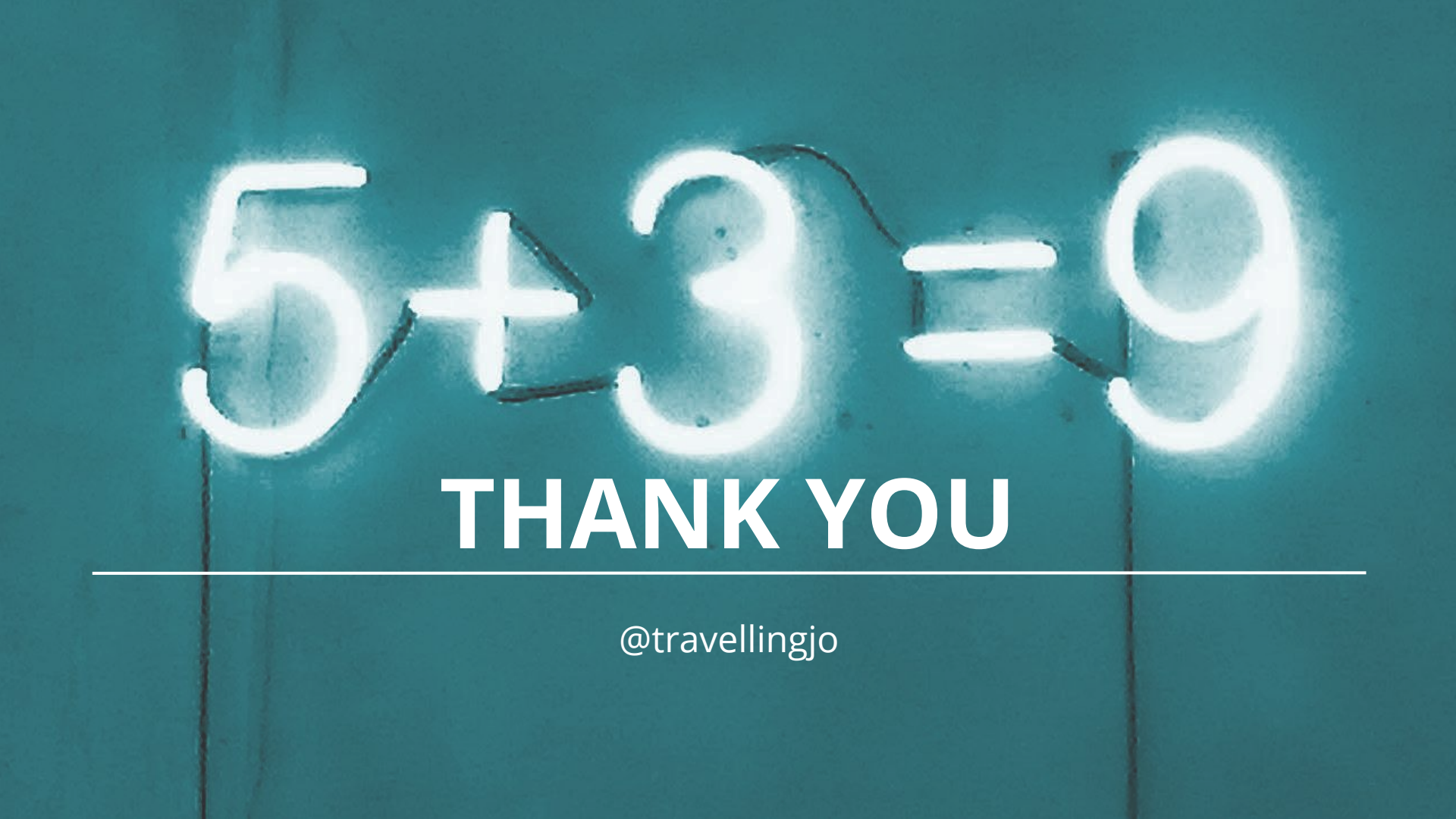
Links

Official Website:
http://galenframework.com/

Galen on Github:
https://github.com/galenframework/galen

Article by creator of Galen:
https://www.smashingmagazine.com/2016/06/the-art-of-layout-testing-with-galen-framework/

Meetup Galen example on Github:
https://github.com/indianajo/Galen-Example



THANK YOU

@travellingjo