Acquia

EXPERIENCE DIGITAL FREEDOM

Automated Accessibility Testing: Using Pa11y and Continuous Integration

Mike Madison

About

Mike Madison
Manager, Technical Services
Acquia Professional Services
mike.madison@acquia.com

Twitter: mikemadison
Linkedin: mikemadison
Blog: http://mikemadison.net

Acquia

This Session

- How to integrate Pa11y into your Drupal project
- How to execute Pa11y tests during continuous integration
- How Drupal configuration and custom development impacts accessibility
- How to provide default content for accessibility testing
- What sorts of things cannot be automated

Overview

Why Automate Accessibility Testing?

1. You want the best possible experience on your site / platform for all users.

2. You are legally / contractually obligated to do so.

3. You don’t want to get sued.

DISCLAIMER

Pa11y will help with any / all of these goals but is insufficient by itself to accomplish them.

What is Pa11y?

A command-line interface which loads web pages and highlights any accessibility issues it finds. Useful for when you want to run a one-off test against a web page.

https://github.com/pa11y/pa11y
https://pa11y.org/

Acquia

What is Pa11y?

Much like other test runners...

- reviews markup of page
- alerts when markup does not align with given standard(s)

https://github.com/pa11y/pa11y
https://pa11y.org/

Acquia

What is Pa11y?

Warning:

1. Pa11y IS NOT:
    a. a real person
    b. a guarantee

2. Pa11y CAN NOT:
    a. act as a screen reader
    b. act as a keyboard
    c. do more than you tell it to do

Take Away 1

Automated Accessibility Testing isn't a golden ticket.

Acquia

Take Away 2

As with any automated testing framework...

Pa11y tests are only as good as you make them.

Drupal and Accessibility

Believe it or not...

YOU are the thing that is most likely to impact the accessibility of a Drupal site.

Acquia

5 Common Mistakes

1. Improperly Configured WYSIWYG
2. Embedded vs. File Field vs. Media Entity
3. Isolated Design Process
4. Requirements Communication
5. Markup Only Testing

https://mikemadison.net/blog/2020/9/22/5-common-drupal-9-accessibility-mistakes

Acquia

Goals

- Anytime you build something, check the accessibility of that thing
  - Lighthouse, Pa11y, etc.
- Fix the accessibility issue(s) found
- “Write a Pa11y Test” to ensure the issue(s) don’t regress

Writing Pa11y Tests

```
/**
 * @file
 * Pa11y config.
 */

const isCI = process.env.CI;
const baseURL = isCI ? 'http://127.0.0.1:8888' : 'http://drupalgovcon.lndo.site:8080';

// Add urls for a11y testing here.
const urls = [
    '/',
];

module.exports = {
  defaults: {
    standard: 'WCAG2AA',
    hideElements: ['svg'],
    ignore: ['notice', 'warning'],
    chromeLaunchConfig: {
      args: ['--no-sandbox']
    }
  },
  urls: urls.map(url => `${baseURL}${url}`)
```


Fin

Just Kidding.

What are we actually running Pa11y on?

Acquia

Structuring Tests

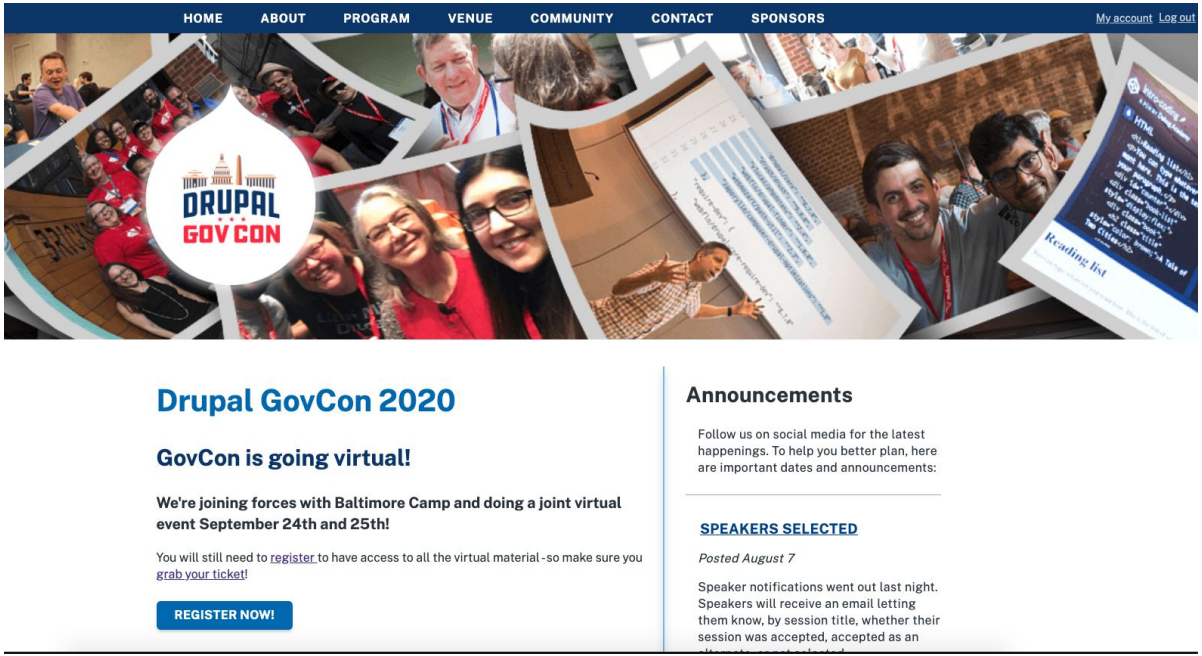
What Pa11y Needs

1. Markup
2. Content
3. Config
4. Drupal
5. Webserver / Database Server
6. Container

Testing a Given Page (e.g. Homepage)

- Proper Theme
- Assets in Place
- Components Placed
  - Announcements Block
  - Header Menu
  - Footer Menu
  - Social Media
  - Sponsors
- Content Published
  - Announcements
  - Homepage Content
  - Hero Banner
  - Main Menu

https://www.drupalgovcon.org/



General Drupal Automated Testing Guidelines

1. Do not rely on a database sync
2. Build everything from a clean install
3. Import your configuration
4. Create content as part of the build

Pa11y Testing Guidelines

Approach 1:

- replicate key pages / features from site

Approach 2:

- create representative content

Component Based Approach

Component-based software engineering (CBSE), also called components-based development (CBD), is a branch of software engineering that emphasizes the separation of concerns with respect to the wide-ranging functionality available throughout a given software system.

It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems.

https://en.wikipedia.org/wiki/Component-based_software_engineering

Consider Your Content and Theme Architecture

Try to make each content bundle on your site as agnostic as possible.

Can you abstract / break up your pages into components on a page?

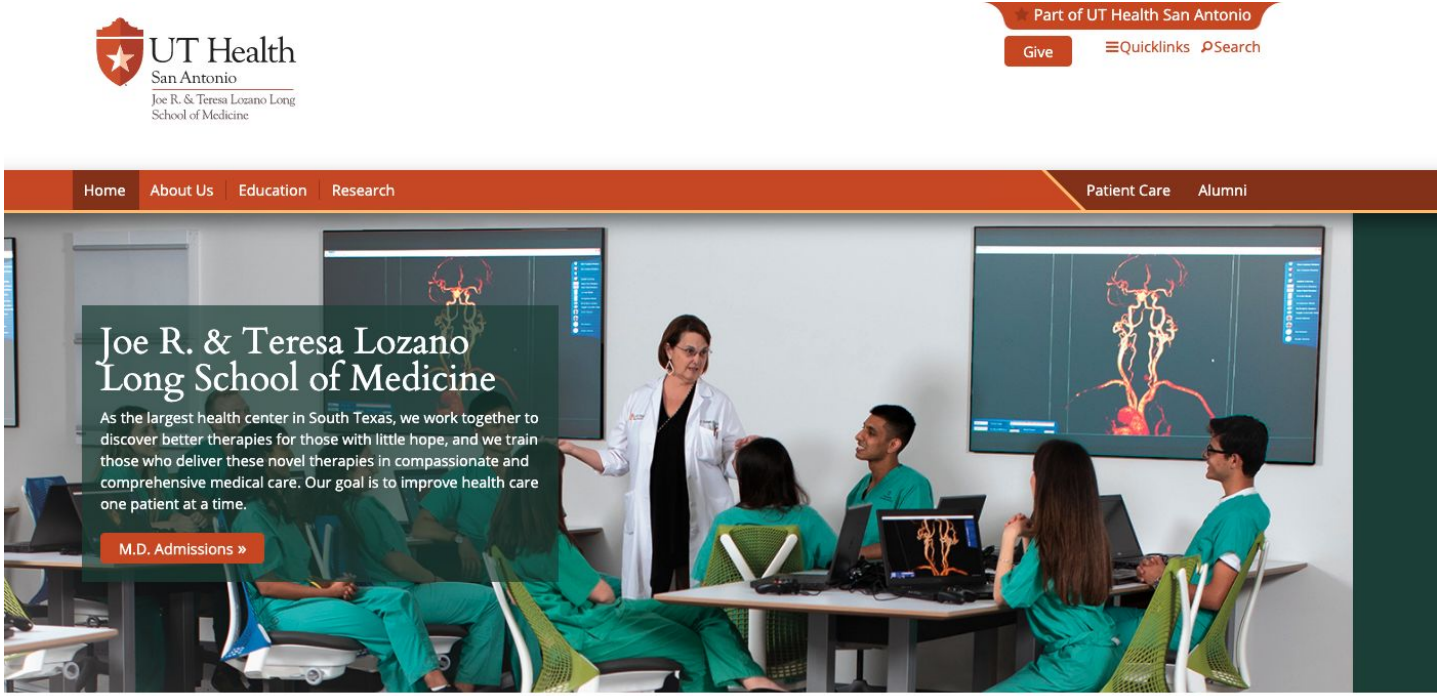
Example

Content:

- 10 Content Types
- 5 Media Types
- 25 Custom Block Types
- 10 View Pages
- 5 View Blocks

Example in Practice

https://www.uthscsa.edu/academics/medicine



Acquia

Testing Our Example

1. Visit one of each content type
    a. make sure media is represented
2. Visit each view page
    a. make sure each is populated
3. Create “testing pages”
    a. remaining view blocks
    b. remaining custom blocks

Testing Our Example

1. Confirm that each component is rendering properly
    a. and each variation of each component
2. Confirm that structured content is rendering properly
3. Confirm that the theme / layout is rendering properly

Reminder: You do not have to test every page of your site in order to accomplish these things!

Take Away 3

Pa11y is not helpful without content and markup!

Acquia

Getting Started

Installing Pa11y For Project Work

https://github.com/Drupal4Gov/Drupal-GovCon-2017/pull/890

1. npm
2. pa11y-ci
3. pa11y configuration
4. web server
5. markup / content

Pa11y vs. Pa11y-CI

Pa11y

A command-line interface which loads web pages and highlights any accessibility issues it finds. Useful for when you want to run a one-off test against a web page.

Pa11y CI

A command-line tool which iterates over a list of web pages and highlights accessibility issues. This is a CLI that's more geared towards use in CI.

When to add Pa11y

Drupal GovCon 2020

https://capitalcampdev.prod.acquia-sites.com/

100

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

Additional items to manually check (10) — These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

Passed audits (21)

Not applicable (20)

Runtime Settings

URL	https://capitalcampdev.prod.acquia-sites.com/
Fetch Time	Sep 14, 2020, 9:50 AM PDT
Device	Emulated Moto G4
Network throttling	150 ms TCP RTT, 1,638.4 Kbps throughput (Simulated)
CPU throttling	4x slowdown (Simulated)

When to Start Accessibility Testing

My “usual” development order for a Content Bundle:

- Site Building Story (builds out the Content Bundle)
- Site Building Story (builds out blocks, views, etc.)
- Backend Story (if required)
- **Theming Story**

All theming work should have accessibility requirements.

Automated accessibility testing should go in place with the theme work.

When to add Pa11y

1. Lighthouse accessibility tests pass
2. Theme churn is minimized
3. Site Building is “done”
4. “Realistic” example content

What does Pa11y Tell You?

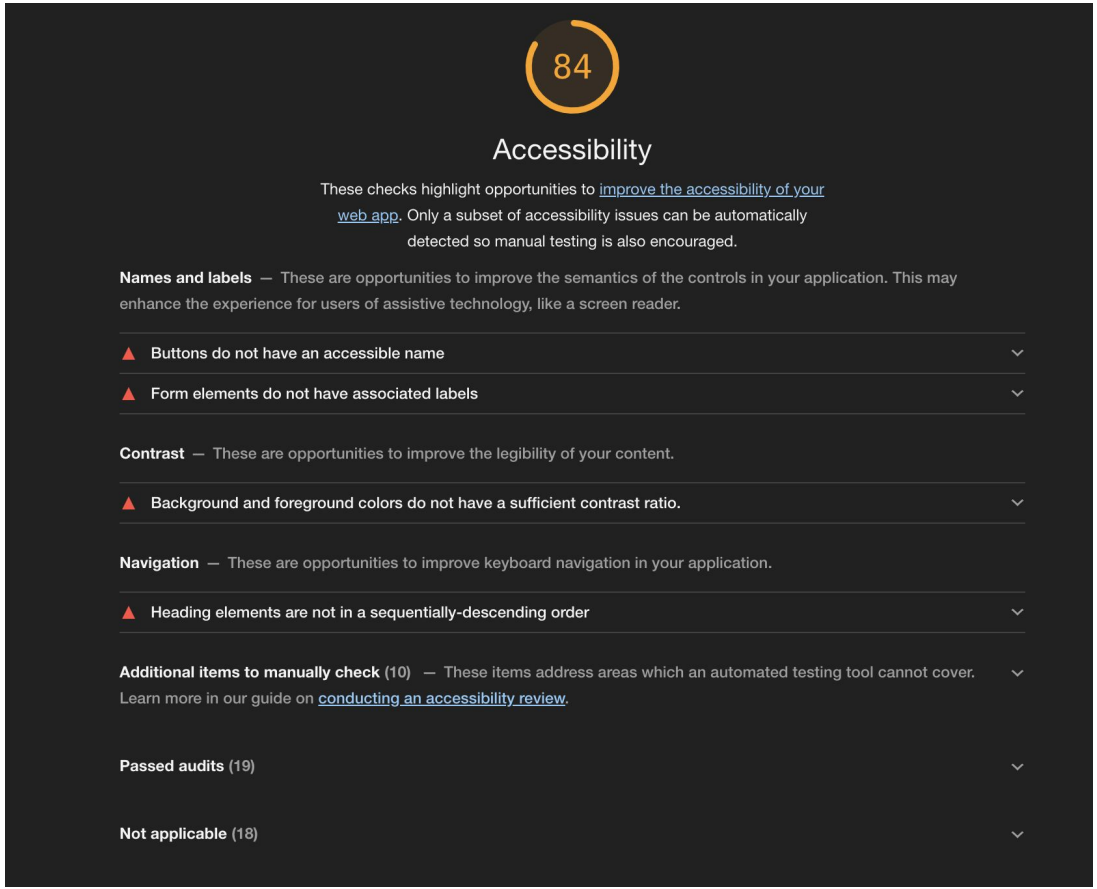
```
http://127.0.0.1:8080/ - 4 errors

Errors in http://127.0.0.1:8080/:

• This link points to a named anchor "main-content" within the document, but
  no anchor exists with that name.
  (html > body > a)
  <a href="#main-content" class="visually-hidden focusable skip-link"> Skip to
  main content </a>
• This textinput element does not have a name available to an accessibility
  API. Valid names are: label element, title undefined, aria-label undefined,
  aria-labelledby undefined.
  (#edit-keywords)
  <input data-drupal-selector="edit-keywords"
  data-search-api-autocomplete-search="search" class="form-autocomplete
  form-text ui-autocomplete-input"
  data-autocomplete-path="/search_api_autocomplete/search?display=search&amp;&amp;filter=keywords"
  type="...
• This form field should be labelled in some way. Use the label element
```

Acquia

What does Pa11y Tell You?



Acquia

What doesn’t Pa11y Tell You?

1. Screen reading
2. Keyboard navigation
3. “Actual” user experience

Configuring Pa11y

.pa11yci.js

```
/**
 * @file
 * Pa11y config.
 */

const isCI = process.env.CI;
const baseURL = isCI ? 'http://127.0.0.1:8888' : 'http://drupalgovcon.lndo.site:8080';

// Add urls for a11y testing here.
const urls = [
    '/',
];

module.exports = {
  defaults: {
    standard: 'WCAG2AA',
    hideElements: ['svg'],
    ignore: ['notice', 'warning'],
    chromeLaunchConfig: {
      args: ['--no-sandbox']
    }
  },
  urls: urls.map(url => `${baseURL}${url}`)
};
```

Acquia

Configuring Pa11y

Reminders:

- the same “test” runs on each page
- adding new “pages” adds new “tests”
- CI process (if using) must:
  - have those pages
  - have content on those pages
  - have styling on those pages

Assumptions

1. You cannot test accessibility without themed content
2. You have some method of creating content in CI
3. You cannot create content without configuration
4. You cannot import configuration with Drupal

How do we do that?

Styling

1. Compile SCSS / JS during the build
    a. This is ideal if you are using NPM / Gulp / Webpack in your build process locally.
    b. Assumes you have gitignored your styles
2. Commit your CSS / JS
    a. Assumes you have not gitignored your styles

https://github.com/Drupal4Gov/Drupal-GovCon-2017/tree/develop/docroot/themes/custom/twentynineteen

How do we do that?

Content

1. Importing Content
    a. setup a CI config split that includes an otherwise disabled module
    b. enable this split during CI (enables module)
    c. module contains an exported set of content using Default Content
2. Creating Content During Build
    a. Caution: most automated testing frameworks (e.g. Behat, PHPUnit) that might create content usually cleanup this content
3. Synchronizing a DB During Build
    a. Caution: databases can change (especially if they are being used for testing or production content). Ideally CI has a consistency that isn't beholden to upstream churn.

https://github.com/Drupal4Gov/Drupal-GovCon-2017/blob/develop/config/default/config_split.config_split.ci.yml

How do we do that?

Configuration

Disclaimer: This isn't a configuration management session!

1. Implement both a configuration management strategy AND configuration management workflow
2. Devise a CI process that takes advantage
3. Ensure that the configuration in CI is representative of the configuration locally / in the cloud.

https://github.com/Drupal4Gov/Drupal-GovCon-2017

Acquia

Putting it all Together

1. Development occurs (whatever that might be)
2. Pull request gets opened
3. Continuous Integration runs
4. Pa11y runs against “current” codebase
5. Validates any / all accessibility tests
6. Fails build if anything regresses***

Warning!

Fails build if anything regresses***

1. This does not mean that a successful build is 100% accessible

2. If a “new feature” gets added w/o configuring the build to test it this new feature hasn’t passed accessibility scanning

Take Away 4

Automating your accessibility testing is the “best” option for regression and monitoring. Every code change should be tested.

Acquia

Other Drupal GovCon Accessibility Content

- The Question Is Moot! Accessibility and Dataviz Is NOT an Either/Or: How to Design Accessible, Usable Data Visualizations

- Mobile accessibility: testing mobile sites and native apps for accessibility

- What can the USA learn from the world on accessibility policy?

Stay in Touch!

Mike Madison

mike.madison@acquia.com

Twitter: mikemadison

Linkedin: mikemadison

Blog: http://mikemadison.net

Acquia

Questions