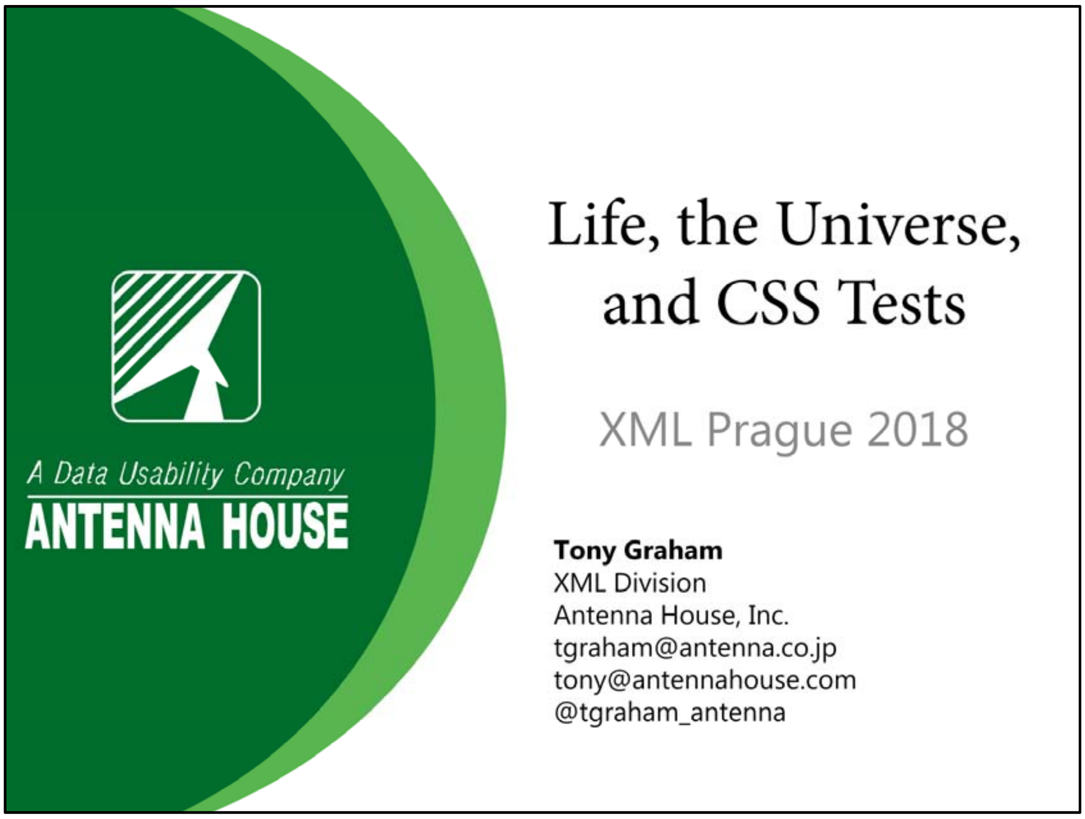


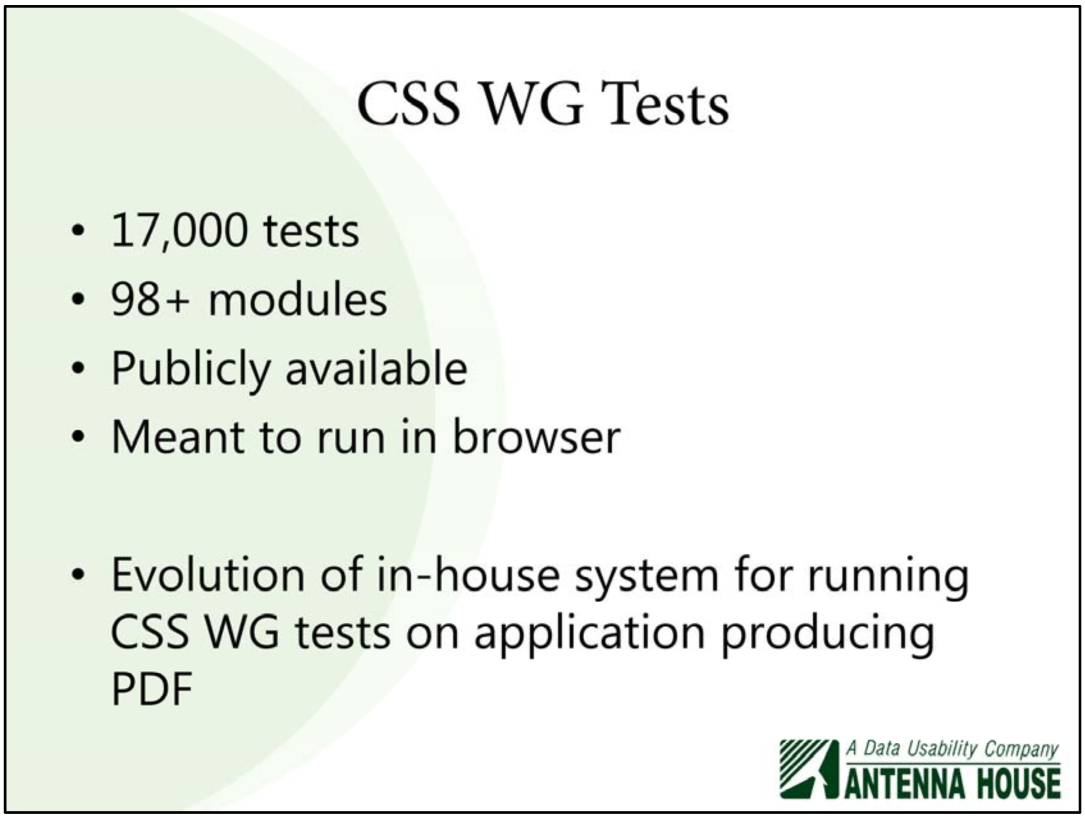
Life, the Universe, and CSS Tests

XML Prague 2018

Tony Graham
XML Division
Antenna House, Inc.
tgraham@antenna.co.jp
tony@antennahouse.com
@tgraham_antenna

It turns out that the answer to the ultimate question of life, the Universe, and CSS Tests isn't a number. It is, in fact, multiple numbers. It is the answers to:

- How many test results are correct?
- How many results in the modules that we care about are correct?
- How many test results have changed in the last AH Formatter build?



CSS WG Tests

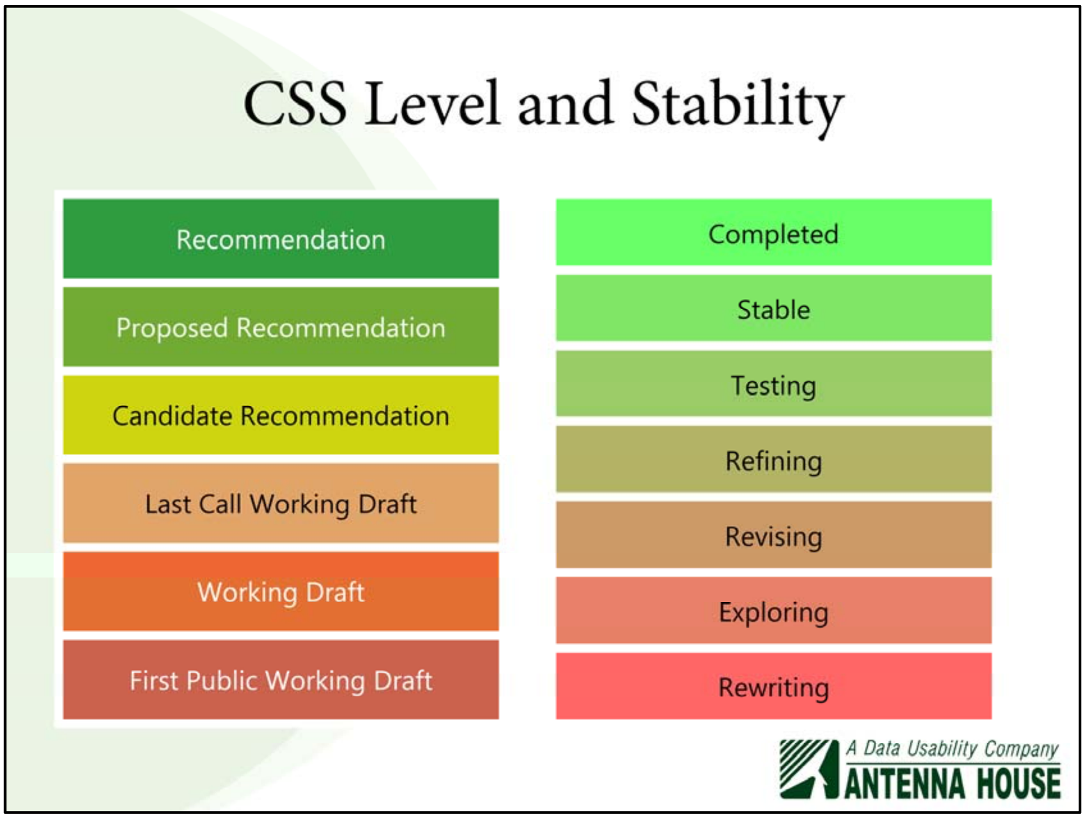
- 17,000 tests
- 98+ modules
- Publicly available
- Meant to run in browser

- Evolution of in-house system for running CSS WG tests on application producing PDF

The tests that I'm talking about is the CSS WG test suite. The CSS Working Group (WG) has a large and ever-expanding set of tests. Currently it contains over 17,000 tests.

The tests are divided into nearly 100 modules (counting CSS 2.1 as multiple modules). The tests are meant to be run in a browser, and the CSS WG provides a browser-based framework for running the tests and comparing them against a reference or, for many tests, checking JavaScript assertions about the test.

However, a browser-based framework and JavaScript assertions aren't much use for a batch formatter that produces PDF, so this talk is about the evolution of an in-house system for running AH Formatter on the CSS WG tests and working with the results.



CSS Level and Stability

Recommendation	Completed
Proposed Recommendation	Stable
Candidate Recommendation	Testing
Last Call Working Draft	Refining
Working Draft	Revising
First Public Working Draft	Exploring
	Rewriting

A Data Usability Company
ANTENNA HOUSE

I’m not going to begin to explain the CSS modules to you. I will say, however, that the CSS WG categorizes their modules by specification level and stability.

Most of you will be familiar with the progression of a W3C specification from ‘First Public Working Draft’ through to ‘Recommendation’.

The stability levels are used only by the CSS WG. The levels range in increasing stability from ‘Rewriting’ and ‘Exploring’ through to ‘Stable’ and ‘Completed’. W3C Recommendations, by definition, are ‘Completed’, but the stability of modules at other levels can vary a lot.

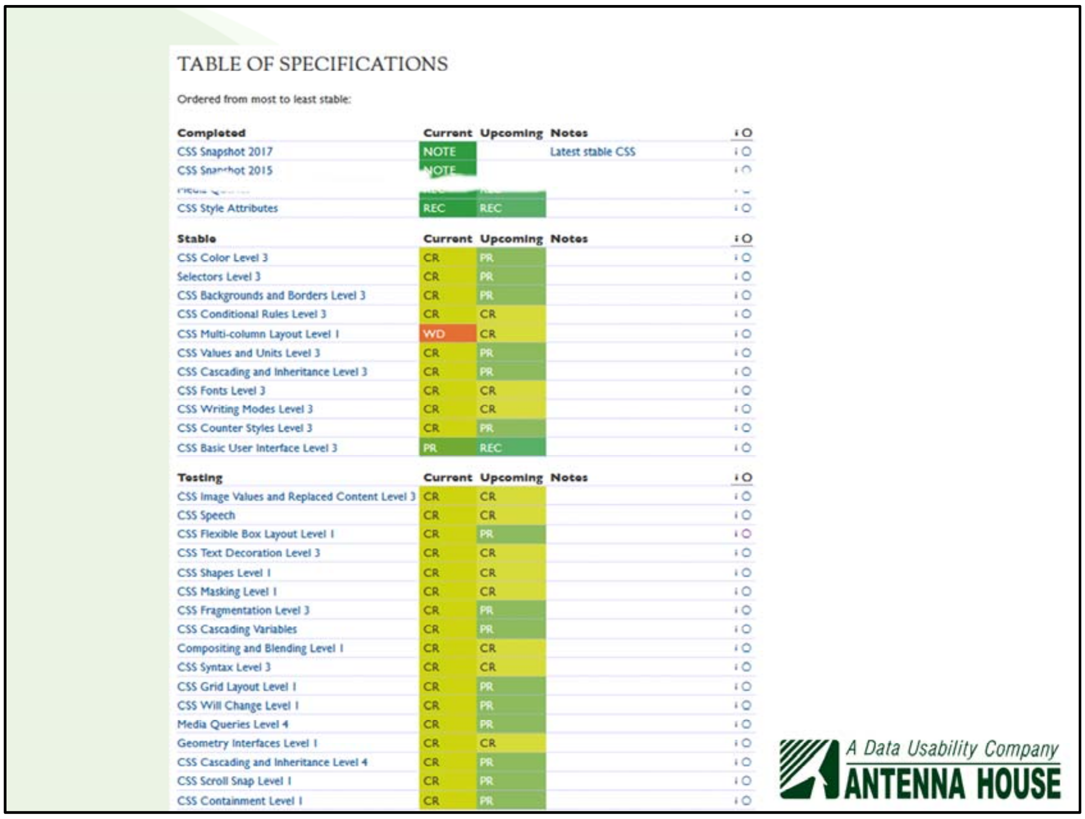


TABLE OF SPECIFICATIONS

Ordered from most to least stable:

Completed	Current	Upcoming	Notes	i ○
CSS Snapshot 2017	NOTE		Latest stable CSS	i ○
CSS Snapshot 2015	NOTE			i ○
Media Queries	[illegible]	[illegible]		i ○
CSS Style Attributes	REC	REC		i ○

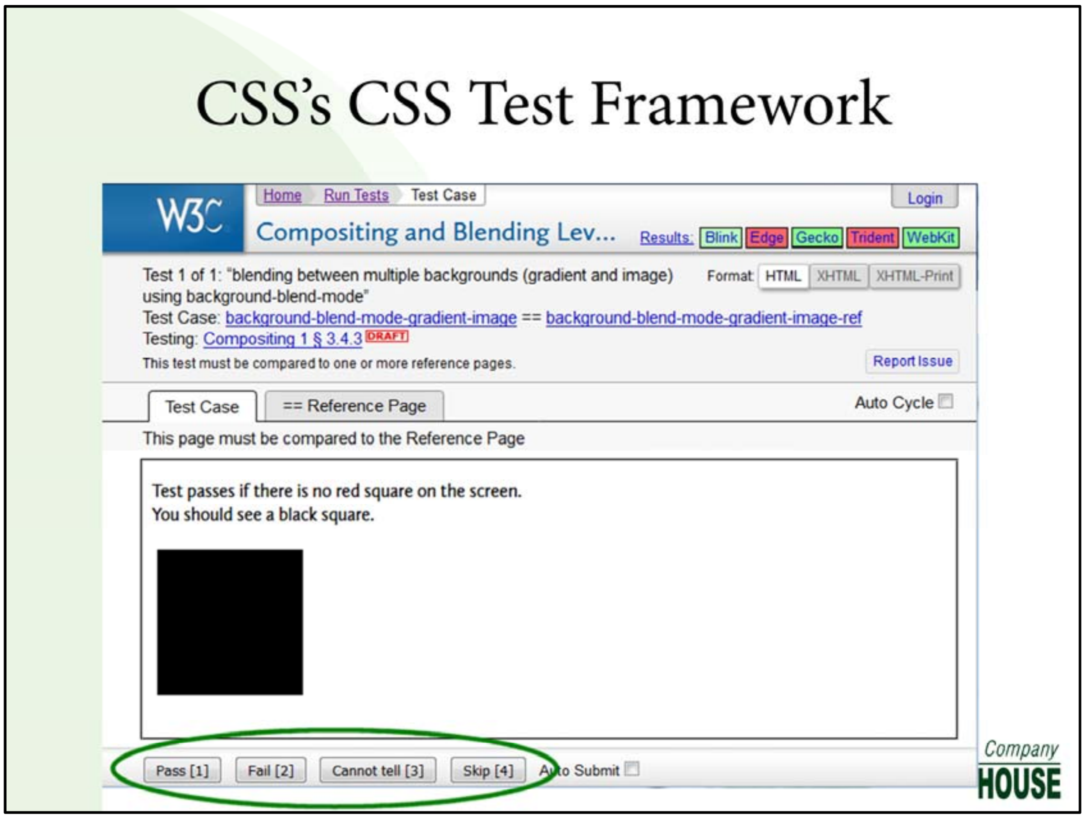
Stable	Current	Upcoming	Notes	i ○
CSS Color Level 3	CR	PR		i ○
Selectors Level 3	CR	PR		i ○
CSS Backgrounds and Borders Level 3	CR	PR		i ○
CSS Conditional Rules Level 3	CR	CR		i ○
CSS Multi-column Layout Level 1	WD	CR		i ○
CSS Values and Units Level 3	CR	PR		i ○
CSS Cascading and Inheritance Level 3	CR	PR		i ○
CSS Fonts Level 3	CR	CR		i ○
CSS Writing Modes Level 3	CR	CR		i ○
CSS Counter Styles Level 3	CR	PR		i ○
CSS Basic User Interface Level 3	PR	REC		i ○

Testing	Current	Upcoming	Notes	i ○
CSS Image Values and Replaced Content Level 3	CR	CR		i ○
CSS Speech	CR	CR		i ○
CSS Flexible Box Layout Level 1	CR	PR		i ○
CSS Text Decoration Level 3	CR	CR		i ○
CSS Shapes Level 1	CR	CR		i ○
CSS Masking Level 1	CR	CR		i ○
CSS Fragmentation Level 3	CR	PR		i ○
CSS Cascading Variables	CR	PR		i ○
Compositing and Blending Level 1	CR	CR		i ○
CSS Syntax Level 3	CR	CR		i ○
CSS Grid Layout Level 1	CR	PR		i ○
CSS Will Change Level 1	CR	PR		i ○
Media Queries Level 4	CR	PR		i ○
Geometry Interfaces Level 1	CR	CR		i ○
CSS Cascading and Inheritance Level 4	CR	PR		i ○
CSS Scroll Snap Level 1	CR	PR		i ○
CSS Containment Level 1	CR	PR		i ○

A Data Usability Company
ANTENNA HOUSE

This is a partial screenshot of the current CSS WG page that lists the modules ordered by stability level and showing their current and next specification levels.

You might think that a module would progress through all the stability levels on the way to being a completed Recommendation, but, as you can see, more 'Testing'-level modules than 'Stable'-level modules will progress next to Proposed Recommendation.



CSS's CSS Test Framework

Home | Run Tests | Test Case | Login

Compositing and Blending Lev... Results: Blink Edge Gecko Trident WebKit

Test 1 of 1: "blending between multiple backgrounds (gradient and image) using background-blend-mode"

Format: HTML XHTML XHTML-Print

Test Case: background-blend-mode-gradient-image == background-blend-mode-gradient-image-ref

Testing: Compositing 1 § 3.4.3 DRAFT

This test must be compared to one or more reference pages.

Report Issue

Test Case | == Reference Page | Auto Cycle

This page must be compared to the Reference Page

Test passes if there is no red square on the screen.
You should see a black square.

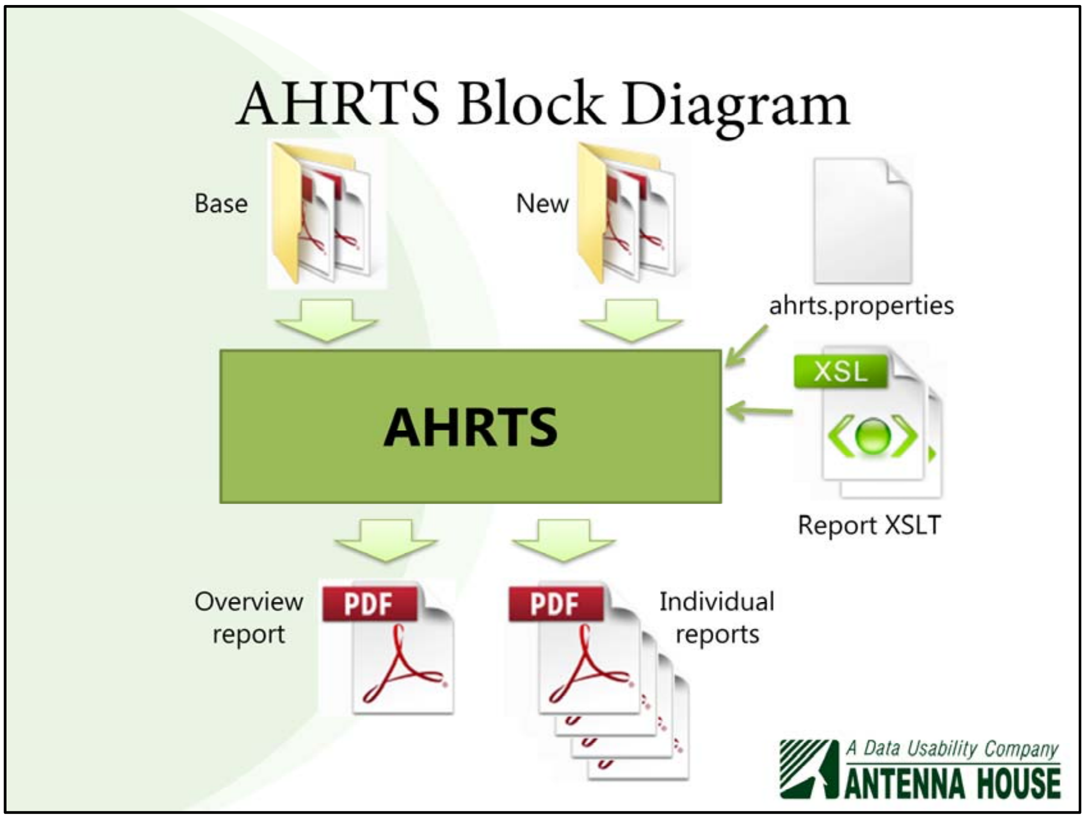
Pass [1] | Fail [2] | Cannot tell [3] | Skip [4] | Auto Submit

This is a screenshot of a CSS test in the CSS WG's test framework. The framework can show the test and its reference page (if there is one).

You can indicate the state of the test result by clicking on one of these buttons.

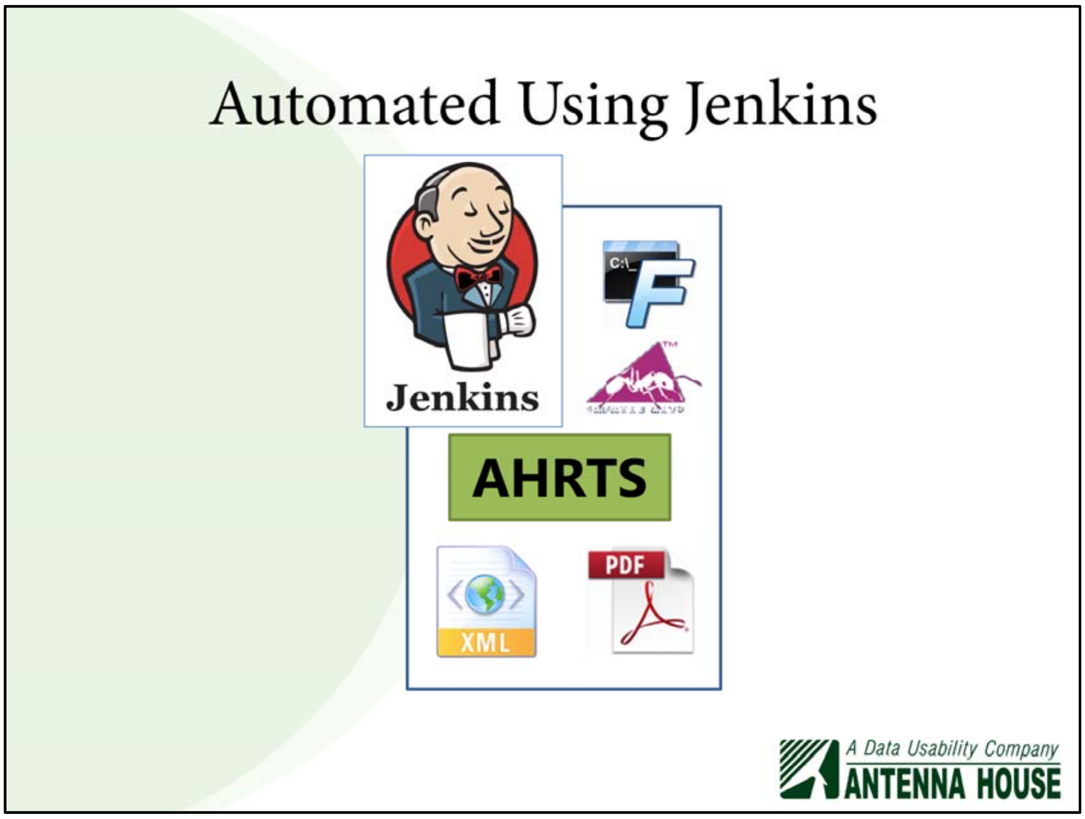
As I said, there's more than 17,000 CSS tests. The CSS Working Group have multiple frameworks for checking test results. Not surprisingly, these run in a browser. Which means, of course, that their framework isn't much use when testing a batch formatter that produces PDF. There really was only one thing to do…

Okay, so there was really two things to do…



AHRTS Block Diagram

I was already using Antenna House’s own regression testing system for testing XSL-FO, so I just adapted that for use with the CSS WG tests.

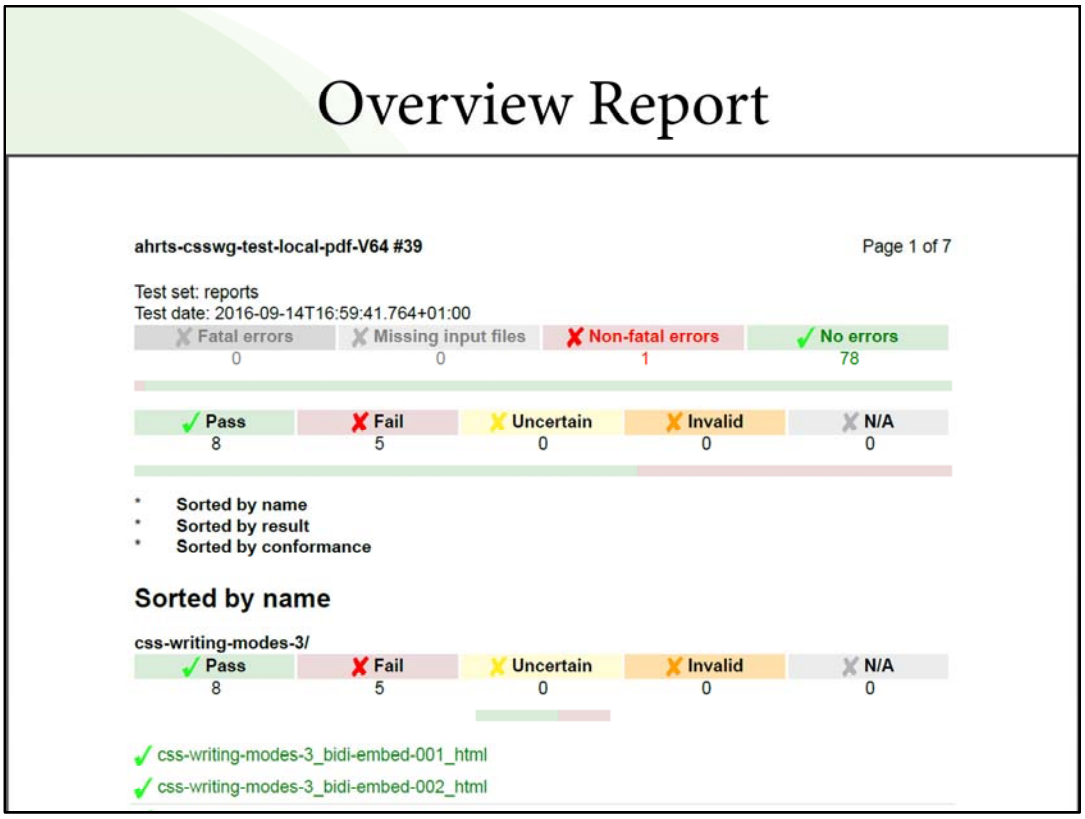


Automated Using Jenkins

I had automated the testing by using the Jenkins Continuous Integration (CI) server.

My Jenkins jobs execute Ant targets so that the Jenkins configuration is as simple as possible.

Ant runs AH Formatter and then runs AHRTS on the results.



Overview Report

ahrts-csswg-test-local-pdf-V64 #39 Page 1 of 7

Test set: reports
Test date: 2016-09-14T16:59:41.764+01:00

Fatal errors	Missing input files	Non-fatal errors	No errors
0	0	1	78

Pass	Fail	Uncertain	Invalid	N/A
8	5	0	0	0

- Sorted by name
- Sorted by result
- Sorted by conformance

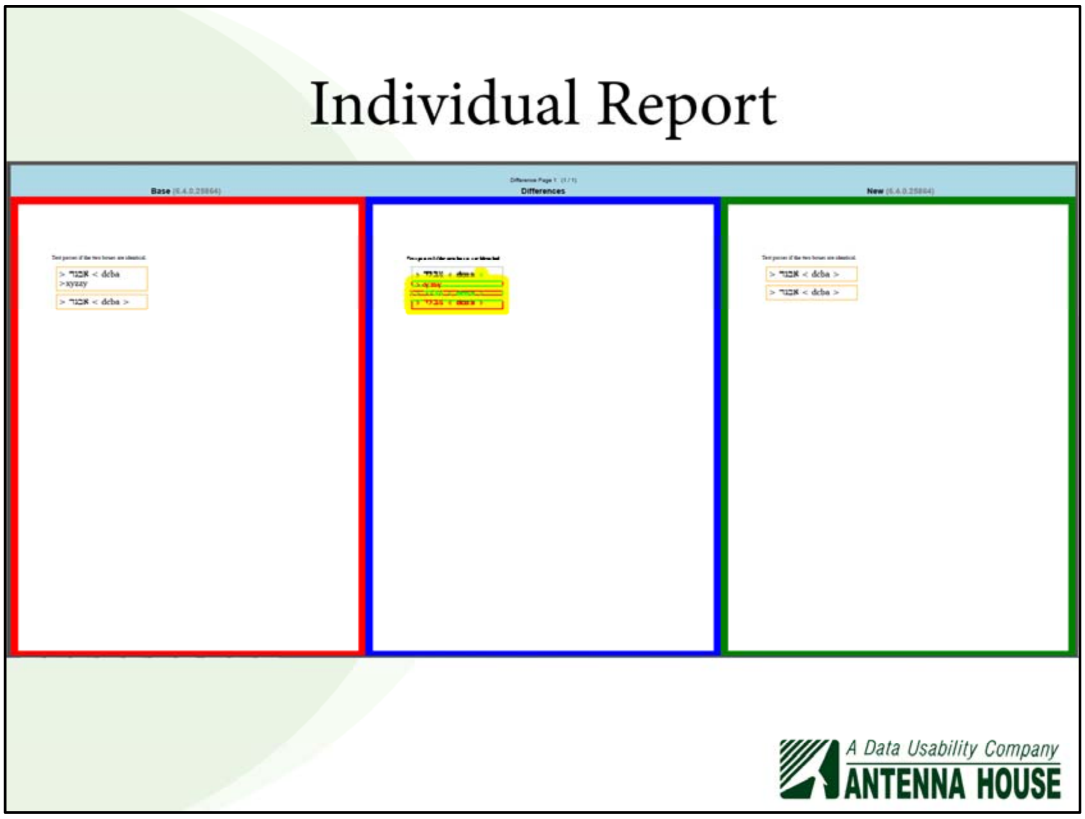
Sorted by name

css-writing-modes-3/

Pass	Fail	Uncertain	Invalid	N/A
8	5	0	0	0

✓ css-writing-modes-3_bidi-embed-001_html
✓ css-writing-modes-3_bidi-embed-002_html

This is an AHRTS overview report from a previous run. It’s a bit different from the standard AHRTS report since, as well as showing whether there are any differences between the base files and the new files, it’s also reporting on the conformance of the results.



Individual Report

Difference Page 1 (1 / 1)

Base (6.4.0.25864)

Differences

New (6.4.0.25864)

Test passes if the two boxes are identical.

> אבגד < dcba
>xyzzy

> אבגד < dcba >

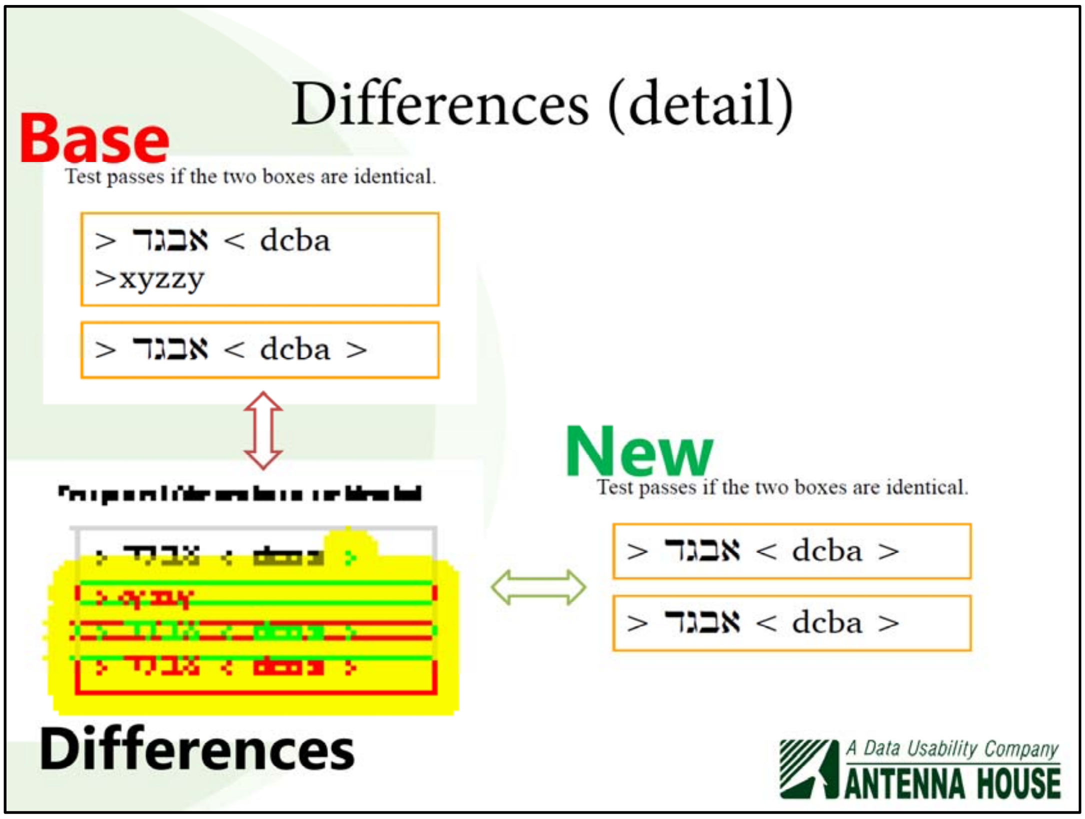
Test passes if the two boxes are identical.

> אבגד < dcba >

> אבגד < dcba >

A Data Usability Company
ANTENNA HOUSE

The individual reports highlight any differences between the base and the new version of a file.



Differences (detail)

Base

Test passes if the two boxes are identical.

> אבגד < dcba
>xyzzy

> אבגד < dcba >

New

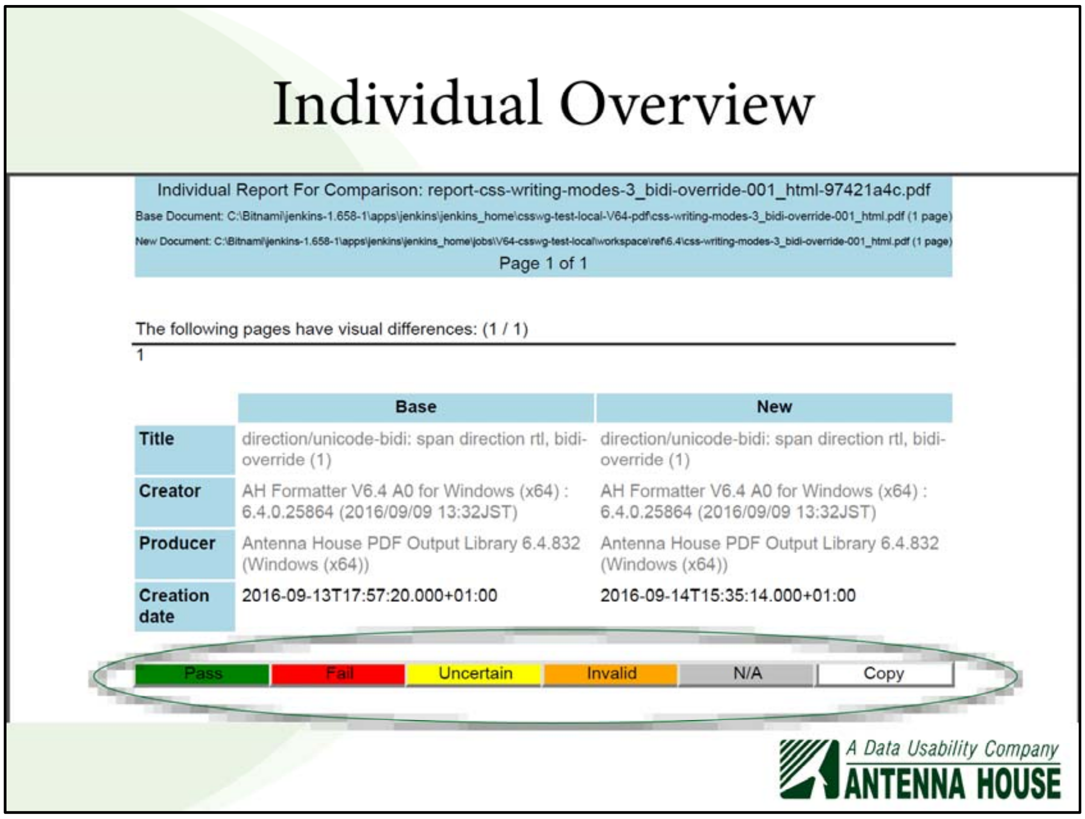
Test passes if the two boxes are identical.

> אבגד < dcba >

> אבגד < dcba >

Differences

To make it easy to see what’s changed, the ‘Differences’ pane has the changes highlighted, with different colors for what’s in one version and what’s in the other.



Individual Overview

Individual Report For Comparison: report-css-writing-modes-3_bidi-override-001_html-97421a4c.pdf

Base Document: C:\Bitnami\jenkins-1.658-1\apps\jenkins\jenkins_home\csswg-test-local-V64-pdf\css-writing-modes-3_bidi-override-001_html.pdf (1 page)

New Document: C:\Bitnami\jenkins-1.658-1\apps\jenkins\jenkins_home\jobs\V64-csswg-test-local\workspace\ref\6.4\css-writing-modes-3_bidi-override-001_html.pdf (1 page)

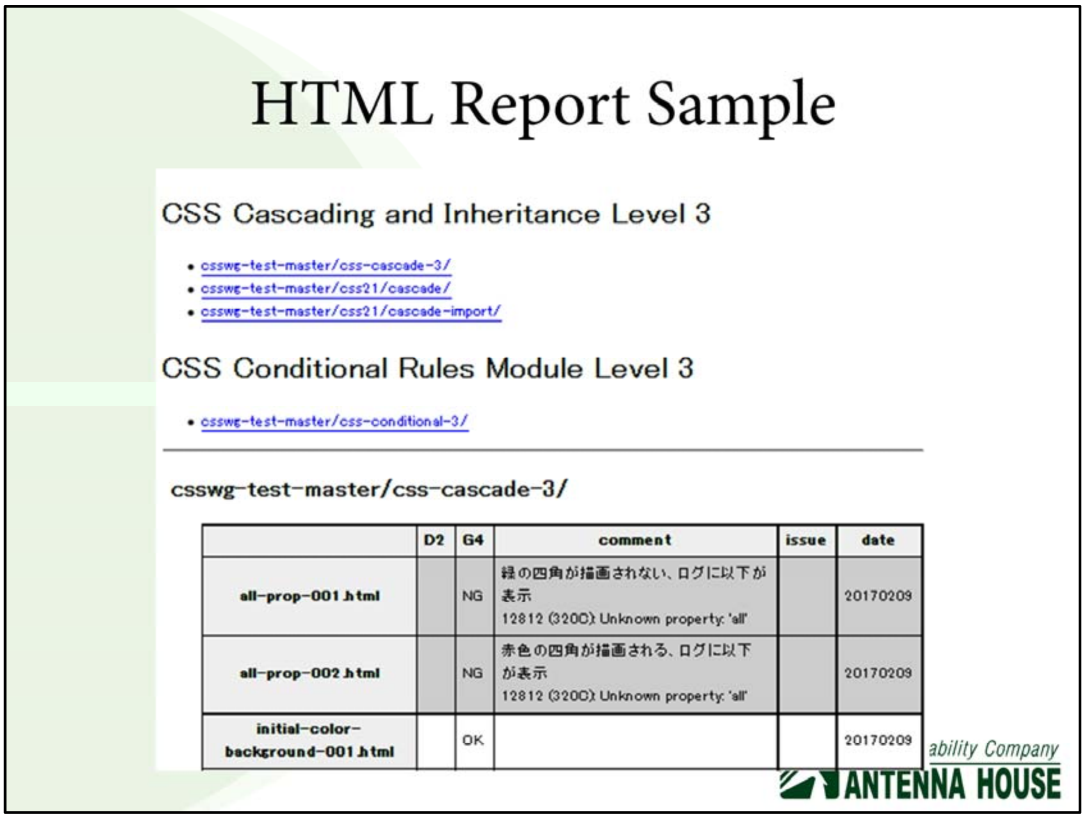
Page 1 of 1

The following pages have visual differences: (1 / 1)

1

	Base	New
Title	direction/unicode-bidi: span direction rtl, bidi-override (1)	direction/unicode-bidi: span direction rtl, bidi-override (1)
Creator	AH Formatter V6.4 A0 for Windows (x64) : 6.4.0.25864 (2016/09/09 13:32JST)	AH Formatter V6.4 A0 for Windows (x64) : 6.4.0.25864 (2016/09/09 13:32JST)
Producer	Antenna House PDF Output Library 6.4.832 (Windows (x64))	Antenna House PDF Output Library 6.4.832 (Windows (x64))
Creation date	2016-09-13T17:57:20.000+01:00	2016-09-14T15:35:14.000+01:00

My change to the individual reports was to add clickable links corresponding to the different states that the CSS WG uses to report test results. Each link runs a Jenkins job that logs the test’s result. The intention was to eventually be able to submit the test results to the CSS WG so more AH Formatter results could be listed alongside the browsers’ results.



HTML Report Sample

CSS Cascading and Inheritance Level 3

- csswg-test-master/css-cascade-3/
- csswg-test-master/css21/cascade/
- csswg-test-master/css21/cascade-import/

CSS Conditional Rules Module Level 3

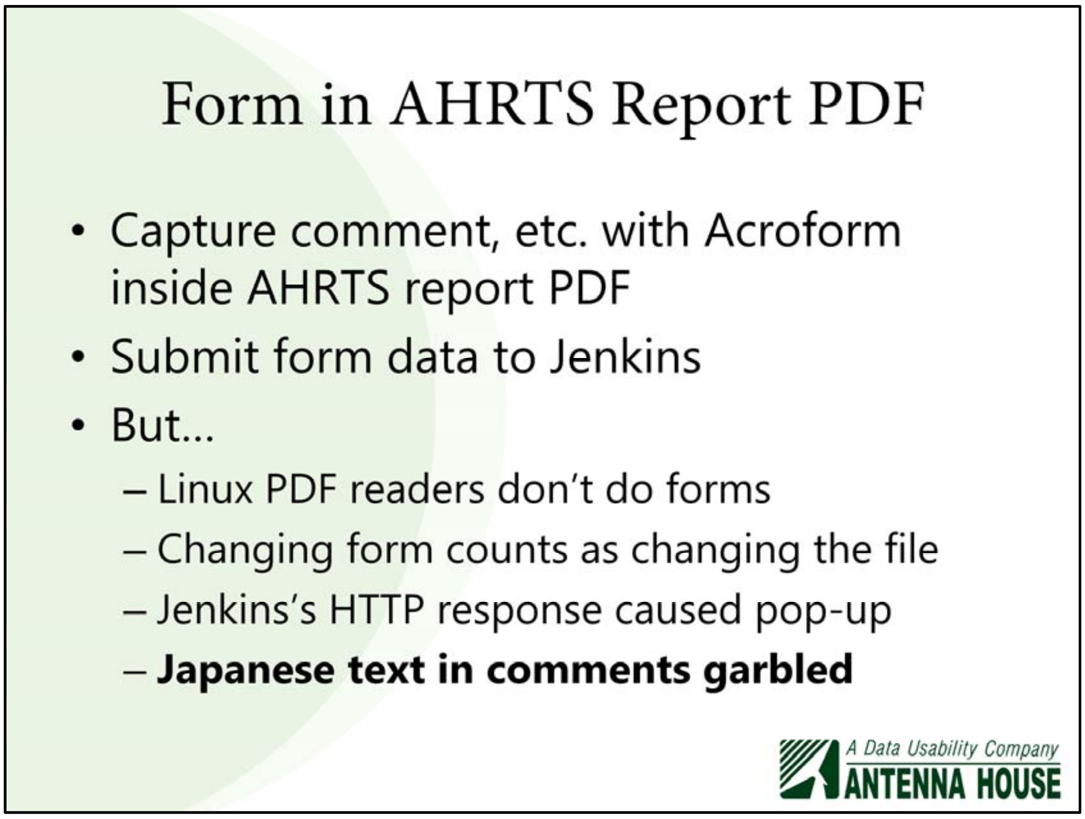
- csswg-test-master/css-conditional-3/

---

csswg-test-master/css-cascade-3/

	D2	G4	comment	issue	date
all-prop-001.html		NG	緑の四角が描画されない、ログに以下が表示 12812 (320C): Unknown property: 'all'		20170209
all-prop-002.html		NG	赤色の四角が描画される、ログに以下が表示 12812 (320C): Unknown property: 'all'		20170209
initial-color-background-001.html		OK			20170209

However, when the developers in Japan looked at the AHRTS report, they wanted a format that was similar to what they'd used when testing MathML and SVG support.



Form in AHRTS Report PDF

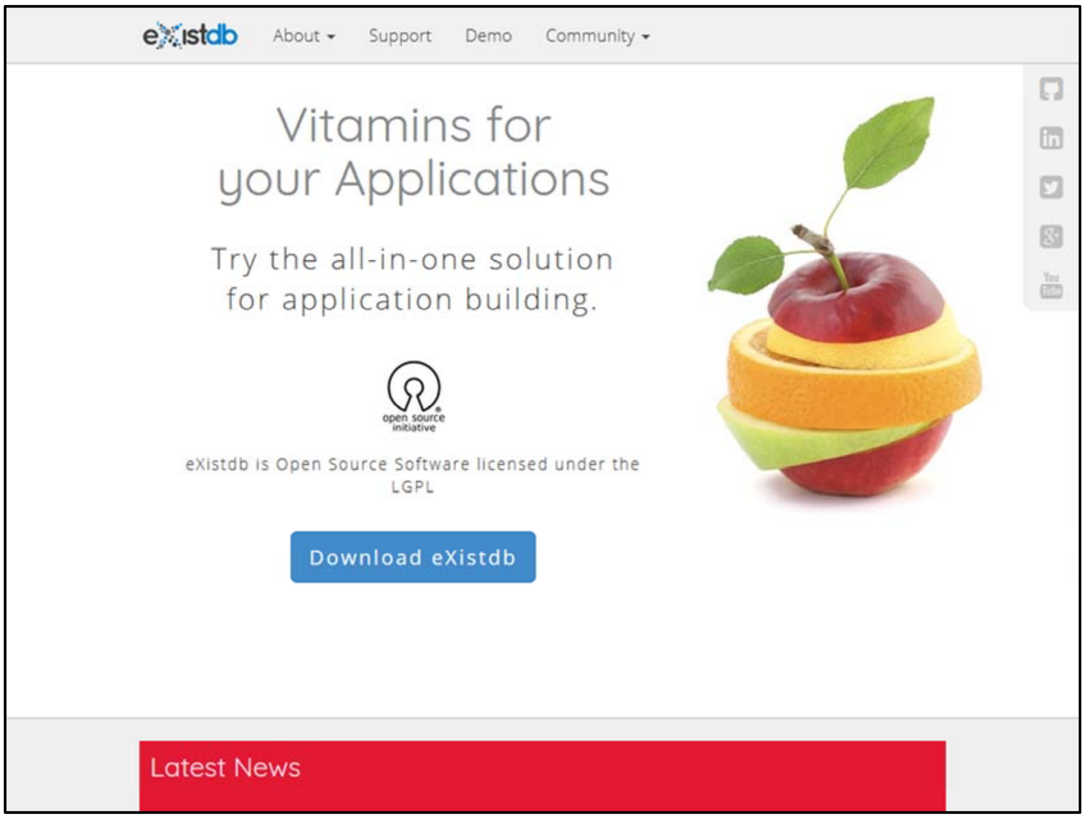
- Capture comment, etc. with Acroform inside AHRTS report PDF
- Submit form data to Jenkins
- But…
  - Linux PDF readers don't do forms
  - Changing form counts as changing the file
  - Jenkins's HTTP response caused pop-up
  - **Japanese text in comments garbled**

A Data Usability Company
ANTENNA HOUSE

This seemed straightforward enough: the extra fields could be captured using an Acroform inside the individual report PDFs that AHRTS produced. The form data would then be submitted to Jenkins.

However, this had several problems:

• I couldn't find a Linux PDF reader that would update the form.
• Acrobat on Windows viewed filling in the form as changing the file, so closing the file brought up a dialog box asking if the changes should be saved. I was later told a workaround for this, but by then I had already moved away from using Acroforms.
• Acrobat would also pop-up a dialog box about storing the HTTP response from Jenkins.
• Lastly and most importantly, Jenkins would correctly store ASCII text from the form fields, but Japanese text was garbled in a way that I could not unscramble.



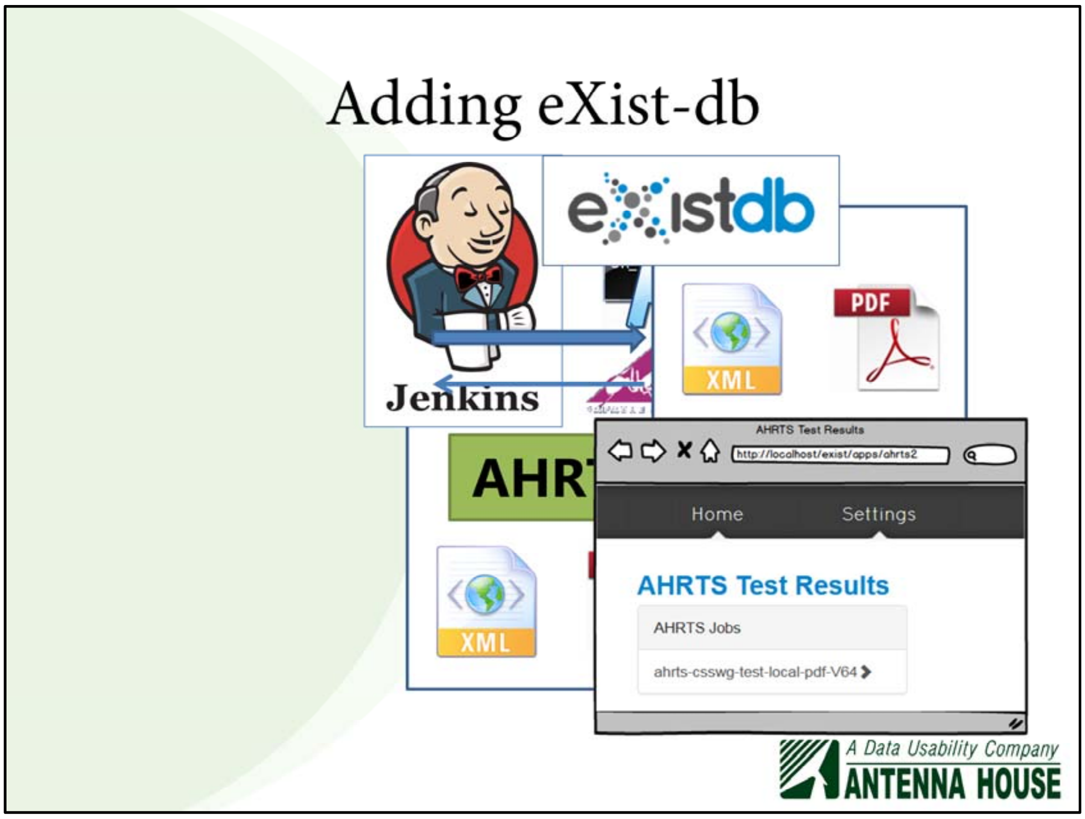
Vitamins for your Applications

Try the all-in-one solution for application building.

eXistdb is Open Source Software licensed under the LGPL

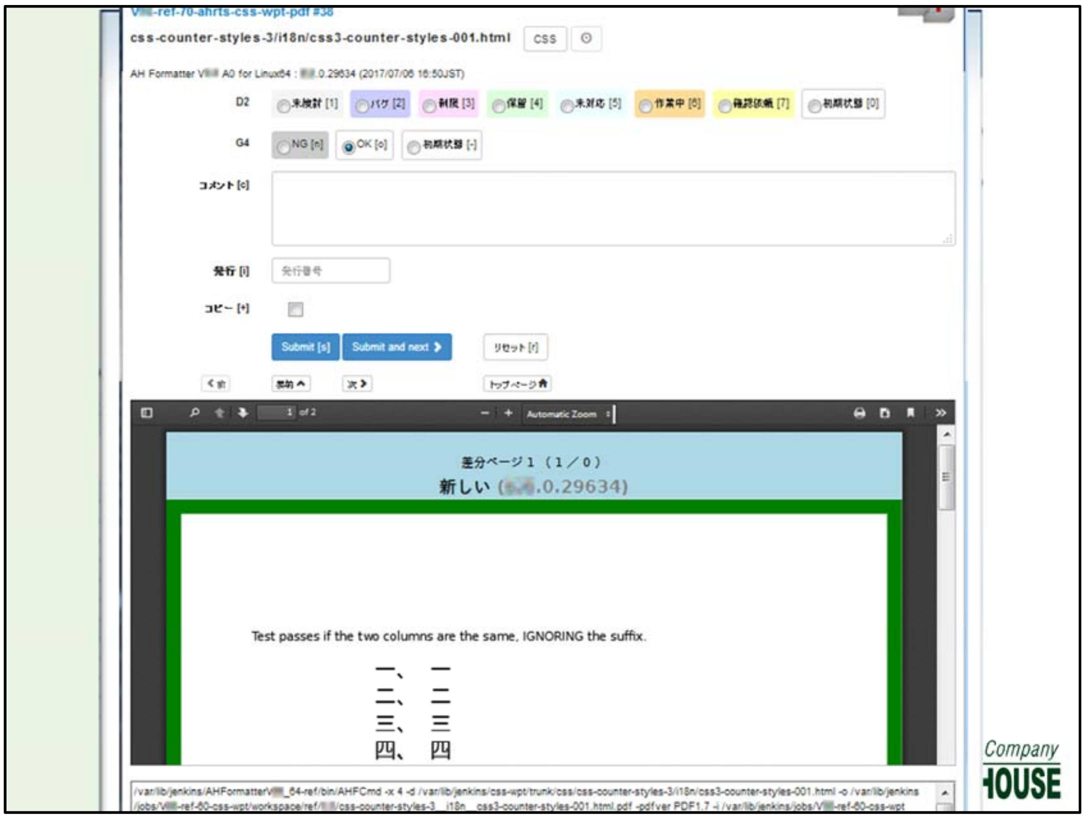
So the project moved to using eXist-db. I had always intended moving to an XML database once the data was complex enough. The data wasn't particularly complex, but the need to handle Japanese text made a good reason to change.

I chose eXist-db partly because I knew some of the developers, which meant that I knew who to ask if I had problems, but the eXist-db mailing list was quite responsive whenever I've had a problem and I've never needed to approach anyone directly.



When eXist-db is added to the picture, Jenkins still runs Ant both to run AH Formatter on the tests and to run AHRTS on the test results. Now, however, the Ant targets also upload both the XML and individual PDFs produced by AHRTS to eXist-db.

This solved the problem with Japanese comments, but it didn’t solve the problems with Acrobat considering modifying the Acroform as modifying the file. The breakthrough came when, instead of putting the form inside the PDF, I put the PDF inside the form and created an eXist-db ‘app’ for viewing the AHRTS results in a web browser.



V[redacted]-ref-70-ahrts-css-wpt-pdf #38

css-counter-styles-3/i18n/css3-counter-styles-001.html CSS

AH Formatter V[redacted] A0 for Linux64 : [redacted].0.29634 (2017/07/06 16:50JST)

D2	○ 未検討 [1] ○ バグ [2] ○ 制限 [3] ○ 保留 [4] ○ 未対応 [5] ○ 作業中 [6] ○ 確認依頼 [7] ○ 初期状態 [0]
G4	○ NG [n] ● OK [o] ○ 初期状態 [-]
コメント [c]	
発行 [i]	発行番号
コピー [+]	☐

差分ページ１（１／０）
新しい ([redacted].0.29634)

Test passes if the two columns are the same, IGNORING the suffix.

一、 一
二、 二
三、 三
四、 四

```
/var/lib/jenkins/AHFormatterV[redacted]_64-ref/bin/AHFCmd -x 4 -d /var/lib/jenkins/css-wpt/trunk/css/css-counter-styles-3/i18n/css3-counter-styles-001.html -o /var/lib/jenkins
/jobs/V[redacted]-ref-60-css-wpt/workspace/ref/[redacted]/css-counter-styles-3__i18n__css3-counter-styles-001.html.pdf -pdfver PDF1.7 -i /var/lib/jenkins/jobs/V[redacted]-ref-60-css-wpt
```

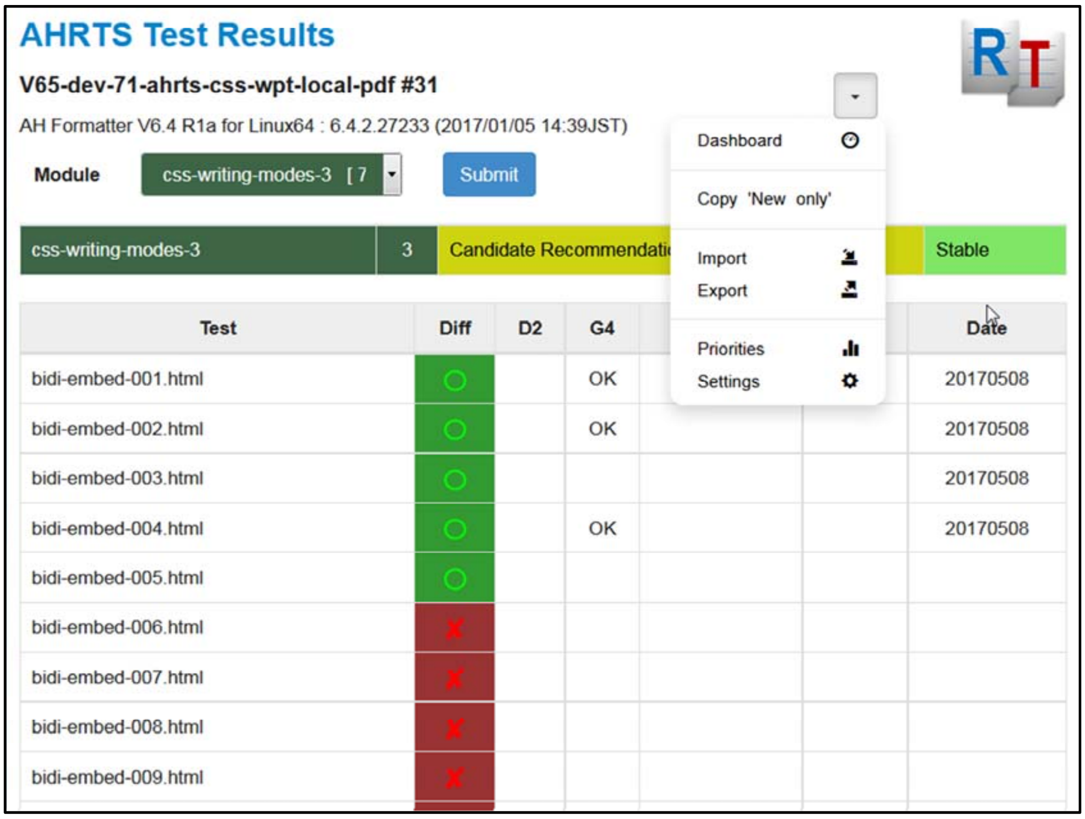
My first version of the app used an XForm for capturing the comment, etc.

The XForm looked rather dated, so I changed the form to use Bootstrap (an open source JavaScript framework).

The single greatest usability improvement in the form was when I added a 'Submit and next' button that both submits the current test's results to eXist-db and advances to show the results for the next test.

The other usability improvement was modifying the XSL-FO for the AHRTS individual PDFs to show the difference page(s) *before* the summary page so that the differences (if any) can be viewed without having to scroll the PDF.

Some tests fail so completely that they don't produce a PDF file, so I added XSLT for Jenkins to run that inserts the logs that AH Formatter generates into the XML reports that AHRTS generates so that eXist-db can include the log in each individual report's HTML page.



AHRTS Test Results

V65-dev-71-ahrts-css-wpt-local-pdf #31

AH Formatter V6.4 R1a for Linux64 : 6.4.2.27233 (2017/01/05 14:39JST)

Module css-writing-modes-3 [7 Submit

css-writing-modes-3	3	Candidate Recommendati		Stable

Test	Diff	D2	G4			Date
bidi-embed-001.html	○		OK			20170508
bidi-embed-002.html	○		OK			20170508
bidi-embed-003.html	○					20170508
bidi-embed-004.html	○		OK			20170508
bidi-embed-005.html	○					
bidi-embed-006.html	✗					
bidi-embed-007.html	✗					
bidi-embed-008.html	✗					
bidi-embed-009.html	✗					

A similar process happened with the summary report.

The initial version was just HTML generated from an XQuery. The first change was that my colleagues in Japan decided that they didn't want to see the totals for each result category.

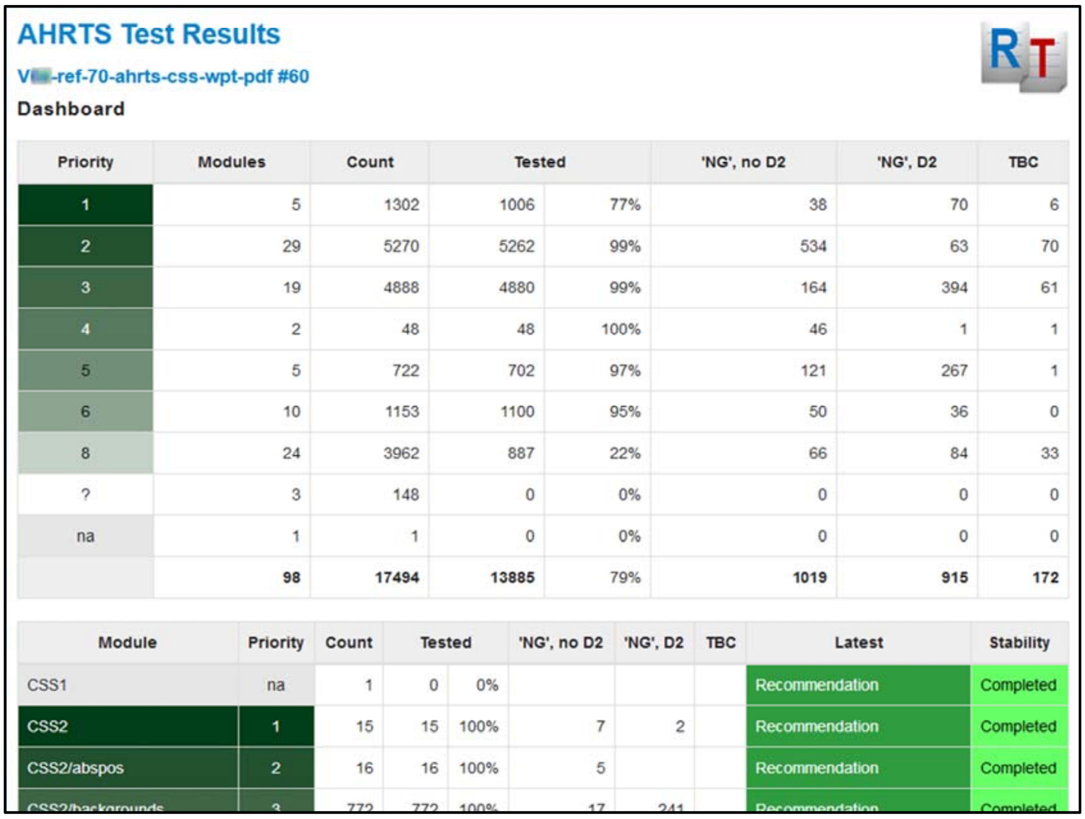
The report was then 'bootstrapped' to improve its appearance.

Then, instead of showing all of the results in one form, the results were delivered module-by-module.

Of course, having decided to not show all of the results all at once, showing all the results at once was added back as an optional view, along with views for other categories of test result.

My colleagues also wanted to see the Antenna House priority for a module plus the specification level and the stability for the module.

I also added options to export and import test results so that results could be shared between my colleagues in Japan and myself.



AHRTS Test Results

V[redacted]-ref-70-ahrts-css-wpt-pdf #60

Dashboard

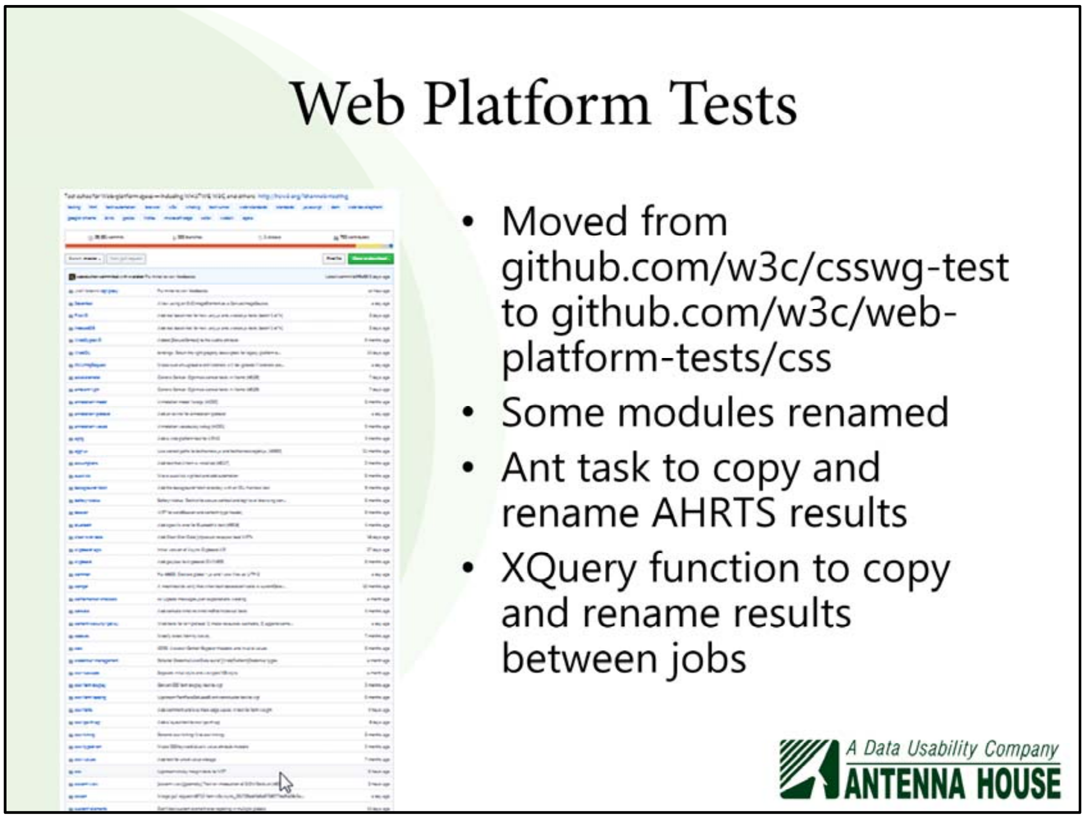
Priority	Modules	Count	Tested		'NG', no D2	'NG', D2	TBC
1	5	1302	1006	77%	38	70	6
2	29	5270	5262	99%	534	63	70
3	19	4888	4880	99%	164	394	61
4	2	48	48	100%	46	1	1
5	5	722	702	97%	121	267	1
6	10	1153	1100	95%	50	36	0
8	24	3962	887	22%	66	84	33
?	3	148	0	0%	0	0	0
na	1	1	0	0%	0	0	0
	98	**17494**	**13885**	79%	**1019**	**915**	**172**

Module	Priority	Count	Tested		'NG', no D2	'NG', D2	TBC	Latest	Stability
CSS1	na	1	0	0%				Recommendation	Completed
CSS2	1	15	15	100%	7	2		Recommendation	Completed
CSS2/abspos	2	16	16	100%	5			Recommendation	Completed
CSS2/backgrounds	3	772	772	100%	17	241		Recommendation	Completed

It turns out that the answer to the ultimate questions of Life, the Universe, and CSS Tests isn't even three numbers. It's actually lots of numbers.

As time went by, my colleagues wanted to see more information about the test results, so I added a dashboard that summarized the test results.

The dashboard started out as a single table, but, over time, it has expanded into three tables, and additional columns have been added to tables whenever someone has needed to know additional information.



Web Platform Tests

- Moved from github.com/w3c/csswg-test to github.com/w3c/web-platform-tests/css
- Some modules renamed
- Ant task to copy and rename AHRTS results
- XQuery function to copy and rename results between jobs

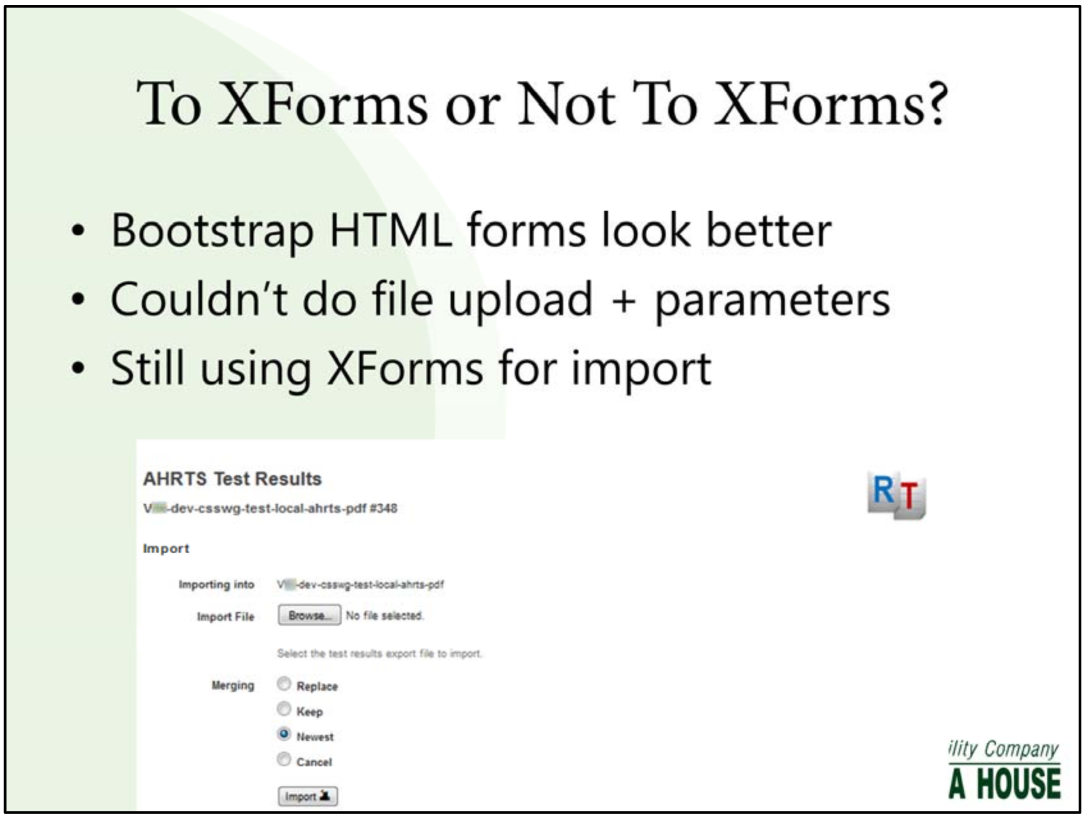
A Data Usability Company
ANTENNA HOUSE

There is a theory which states that if ever anyone has results for the entire CSS test suite, the test suite will instantly be replaced by something even more bizarre and inexplicable. There is another theory that states that this has already happened, since, last year, the CSS WG test suite was merged into the much larger ‘web-platform-tests’ test suite.

This might not have been so bad, but as part of the change, some modules were renamed to align the module name with the short-name of their specification.

This required a one-off Ant task to copy and rename AHRTS test results to make reference files for the new directory structure.

It also required adding an option for a job in eXist-db to copy and rename the job’s results to become the results for the new job for the Web Platform Tests results.



To XForms or Not To XForms?

- Bootstrap HTML forms look better
- Couldn't do file upload + parameters
- Still using XForms for import

AHRTS Test Results

V[redacted]-dev-csswg-test-local-ahrts-pdf #348

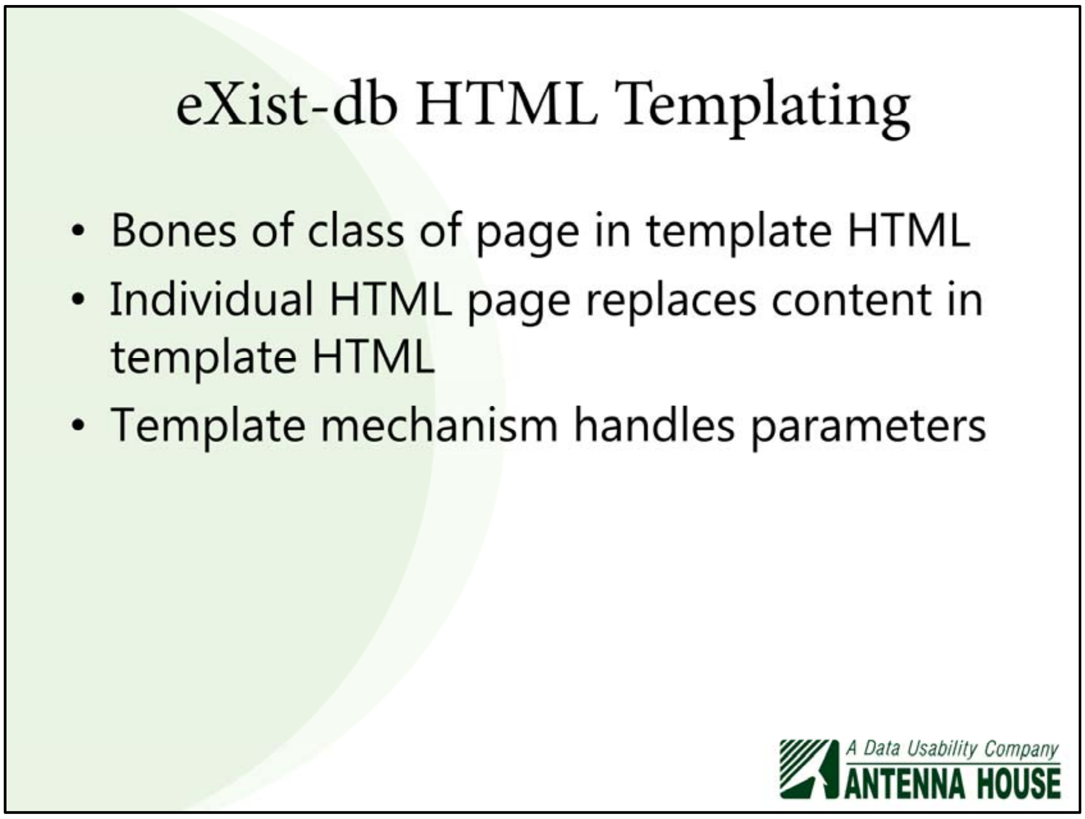
Import

Importing into	V[redacted]-dev-csswg-test-local-ahrts-pdf
Import File	Browse... No file selected.
	Select the test results export file to import.
Merging	Replace
	Keep
	[✓ Newest]
	Cancel
	Import

I'm now going to talk about some aspects of the internals of the project.

When I started using eXist-db, I assumed that I should use XForms with my XML data. However, as you saw before, the XForms pages do not look particularly impressive.

eXist-db uses the Bootstrap JavaScript for its demo application, so I started using Bootstrap with the pages containing forms. I have now replaced all uses of XForms with Bootstrap except for the form for importing test result data which does a file upload.



eXist-db HTML Templating

- Bones of class of page in template HTML
- Individual HTML page replaces content in template HTML
- Template mechanism handles parameters

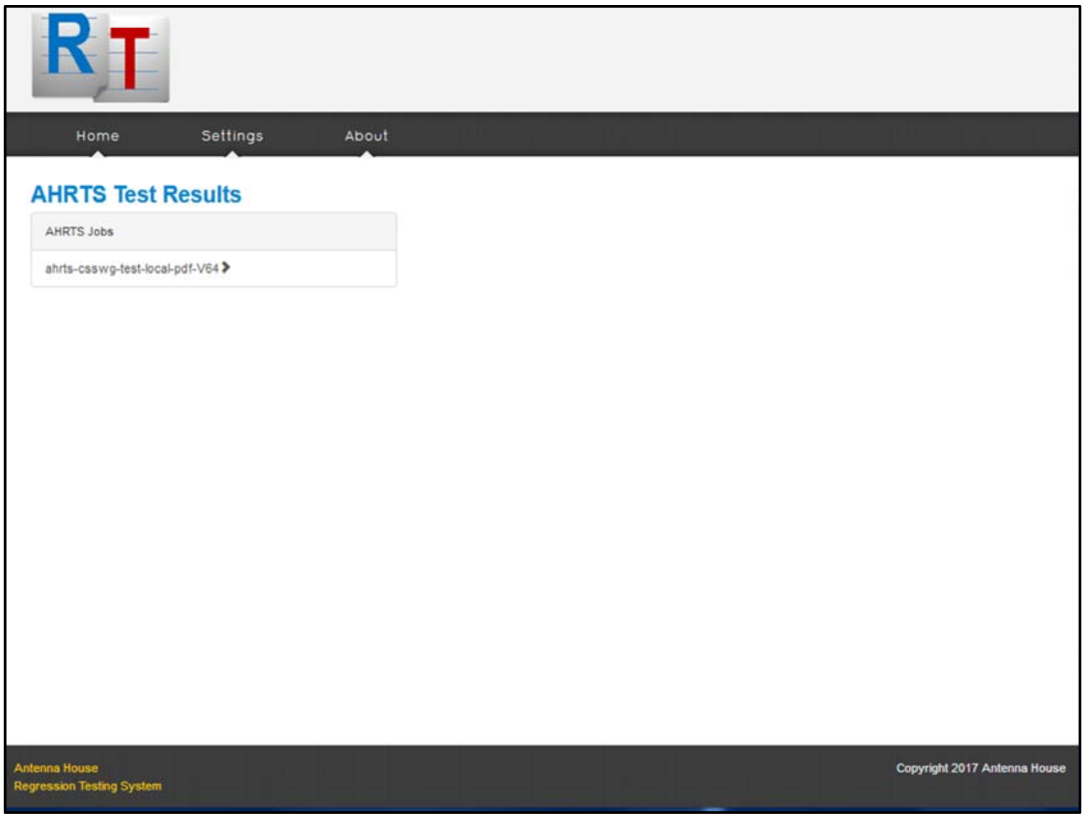
A Data Usability Company
ANTENNA HOUSE

Another feature of the eXist-db demo applications is its templating mechanism.

You create the look of a page or a series of pages using a template HTML file.

The individual HTML page slots into the structure established by the template page.

The templating mechanism handles URI parameters, which further reduces the housekeeping code that you need to write.

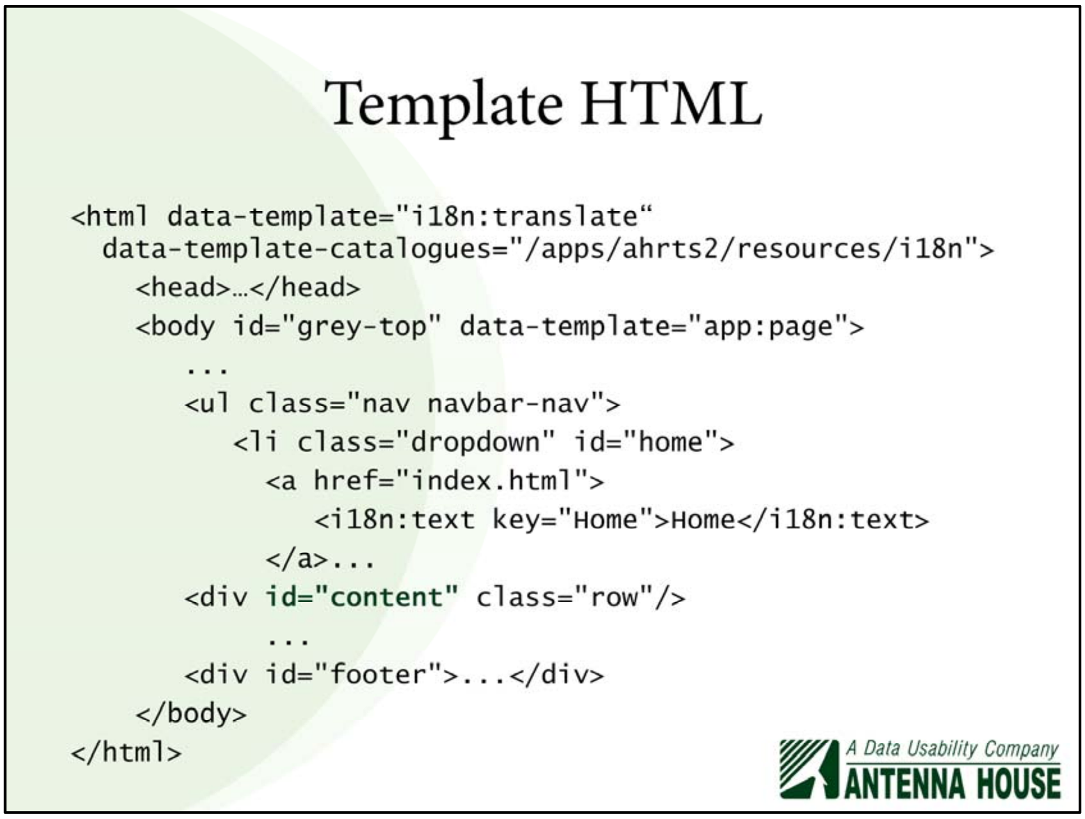


AHRTS Test Results

AHRTS Jobs

ahrts-csswg-test-local-pdf-V64 ❯

This are pages from the app and, as you can see, they are based heavily on the eXist-db sample app.

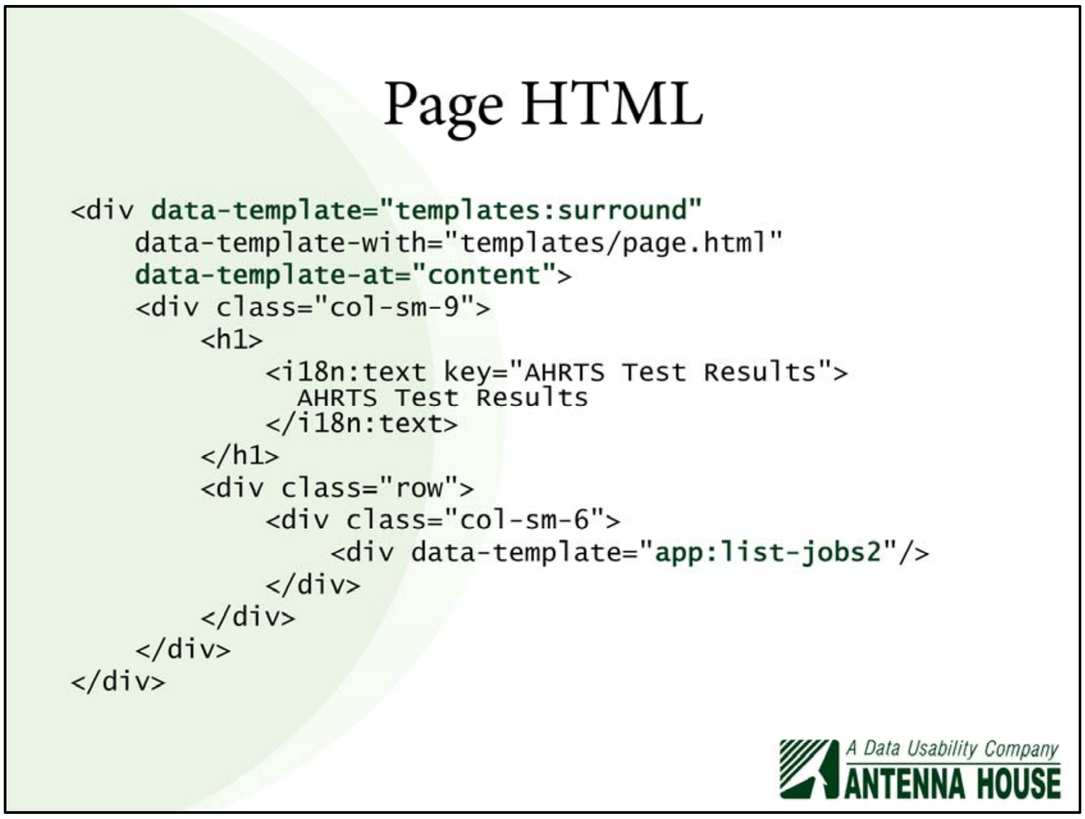


Template HTML

```
<html data-template="i18n:translate“
  data-template-catalogues="/apps/ahrts2/resources/i18n">
    <head>…</head>
    <body id="grey-top" data-template="app:page">
       ...
       <ul class="nav navbar-nav">
          <li class="dropdown" id="home">
            <a href="index.html">
               <i18n:text key="Home">Home</i18n:text>
            </a>...
       <div id="content" class="row"/>
            ...
       <div id="footer">...</div>
    </body>
</html>
```

A Data Usability Company ANTENNA HOUSE

This is a template for a HTML page. Pay attention to the <div> with ‘id=“content“‘.



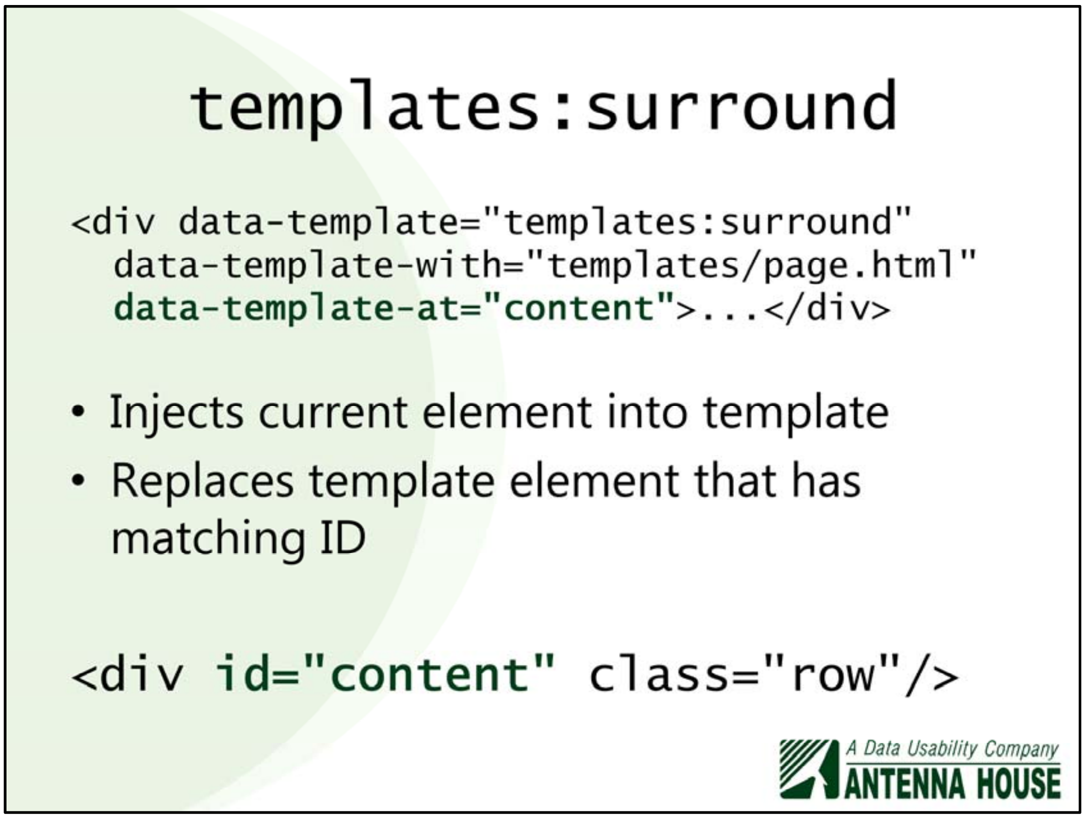
Page HTML

```
<div data-template="templates:surround"
    data-template-with="templates/page.html"
    data-template-at="content">
    <div class="col-sm-9">
        <h1>
            <i18n:text key="AHRTS Test Results">
              AHRTS Test Results
            </i18n:text>
        </h1>
        <div class="row">
            <div class="col-sm-6">
                <div data-template="app:list-jobs2"/>
            </div>
        </div>
    </div>
</div>
```

A Data Usability Company ANTENNA HOUSE

This is the HTML for a page that will be served by the app. This <div> replaces the <div> in the template with the matching ID.

Parts of the HTML are generated by evaluating an XQuery function.



templates:surround

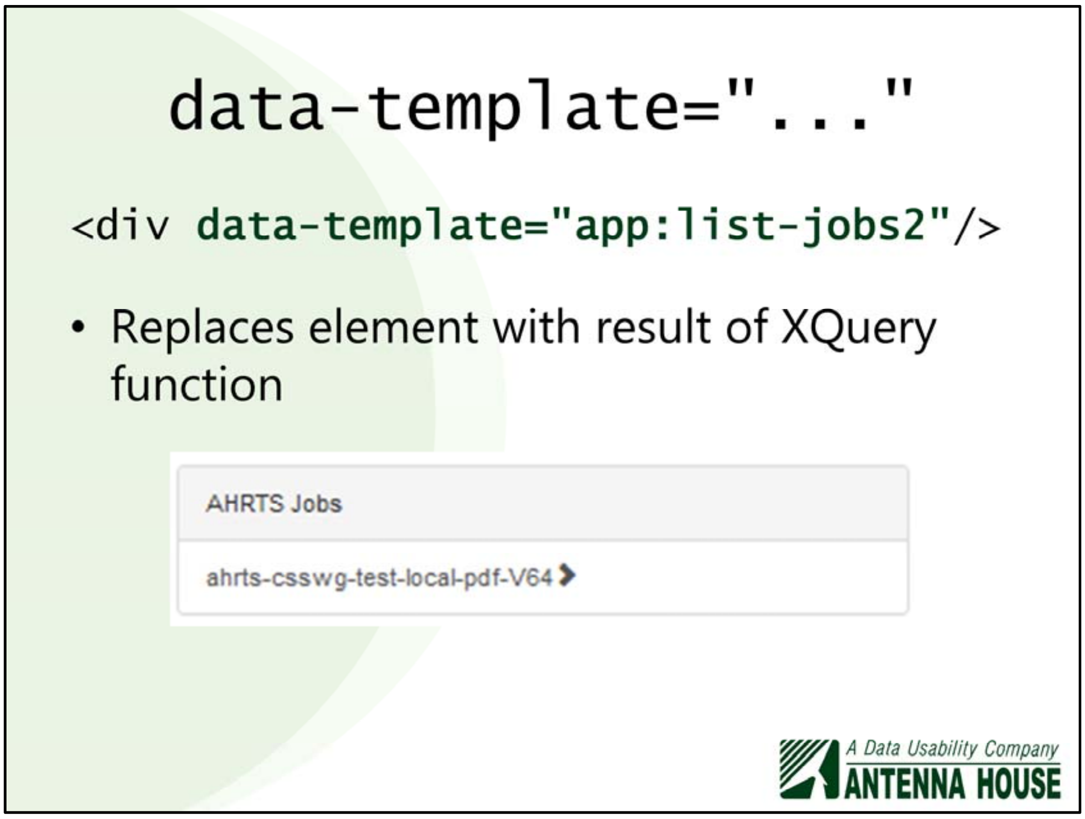
```
<div data-template="templates:surround"
  data-template-with="templates/page.html"
  data-template-at="content">...</div>
```

- Injects current element into template
- Replaces template element that has matching ID

```
<div id="content" class="row"/>
```

This <div> replaces the corresponding <div> from the template when eXist-db serves the HTML page.

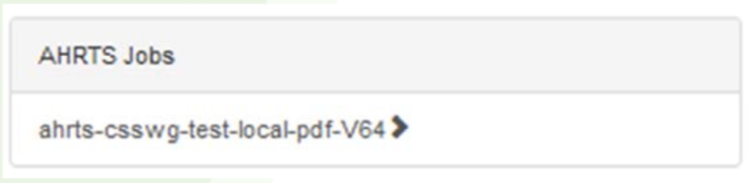
Obviously, different pages of the app can use the same template HTML by providing different content for the <div>.



data-template="..."

```
<div data-template="app:list-jobs2"/>
```

- Replaces element with result of XQuery function

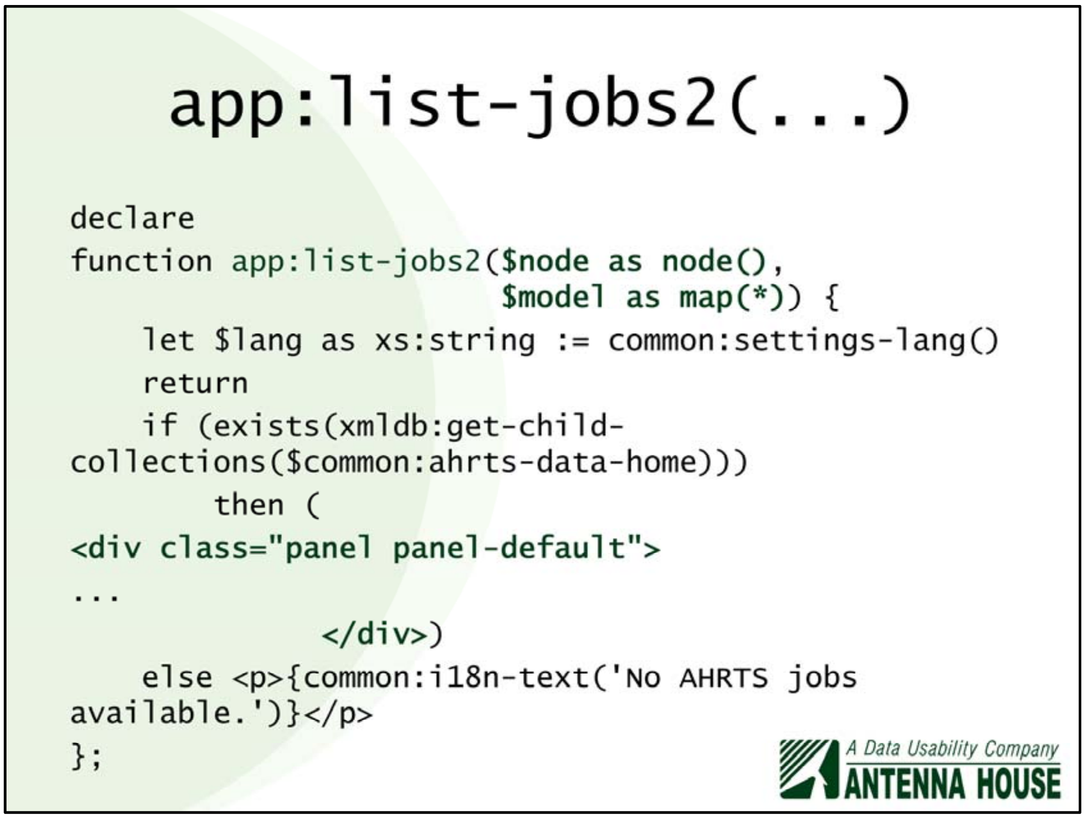


AHRTS Jobs

ahrts-csswg-test-local-pdf-V64 ❯

A HTML element with a ‘data-template’ attribute is replaced by the result of the named function.

In this case, the ‘app:list-jobs2’ function returns the HTML markup for displaying a list of the known jobs.



app:list-jobs2(...)

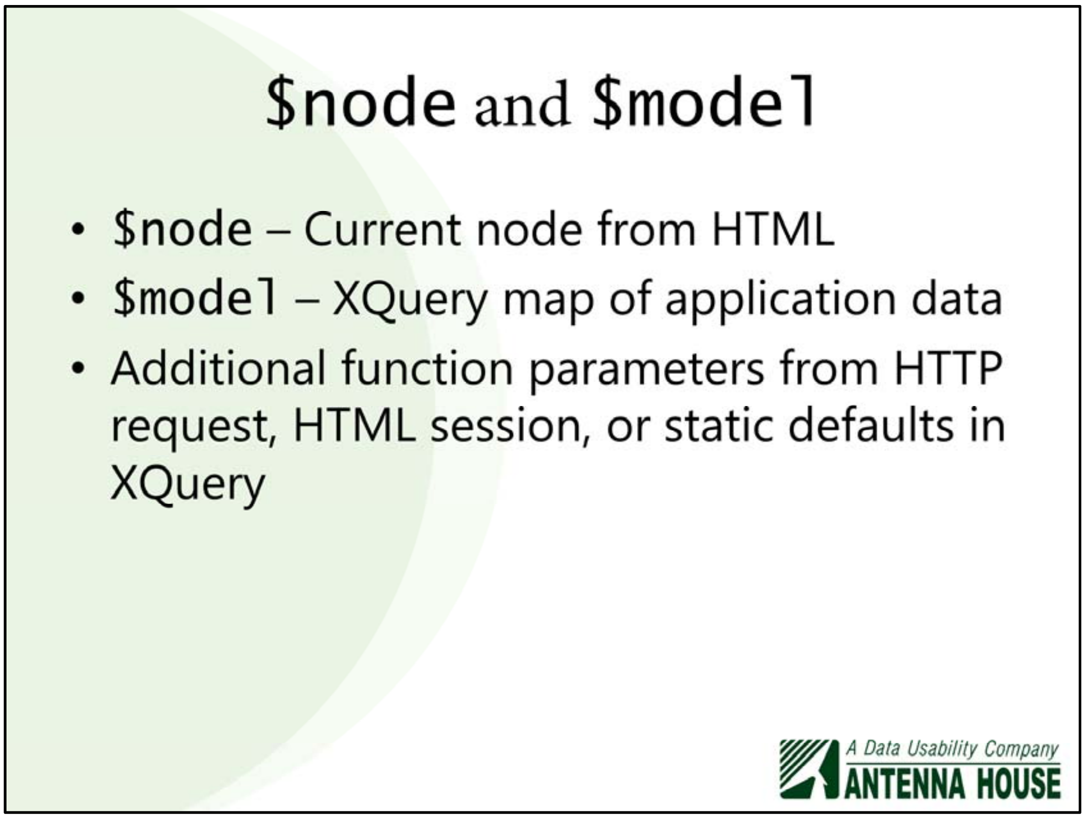
```
declare
function app:list-jobs2($node as node(),
                        $model as map(*)) {
    let $lang as xs:string := common:settings-lang()
    return
    if (exists(xmldb:get-child-
collections($common:ahrts-data-home)))
        then (
<div class="panel panel-default">
...
              </div>)
    else <p>{common:i18n-text('No AHRTS jobs
available.')}</p>
};
```

A Data Usability Company
ANTENNA HOUSE

The ‘app:list-jobs’ function executes the Xquery code for determining the known jobs and returns the HTML markup.

The HTML uses CSS class names from Bootstrap to set the appearance of the HTML.

The ‘$node’ and ‘$model’ arguments are common to every function that the templating mechanism calls.



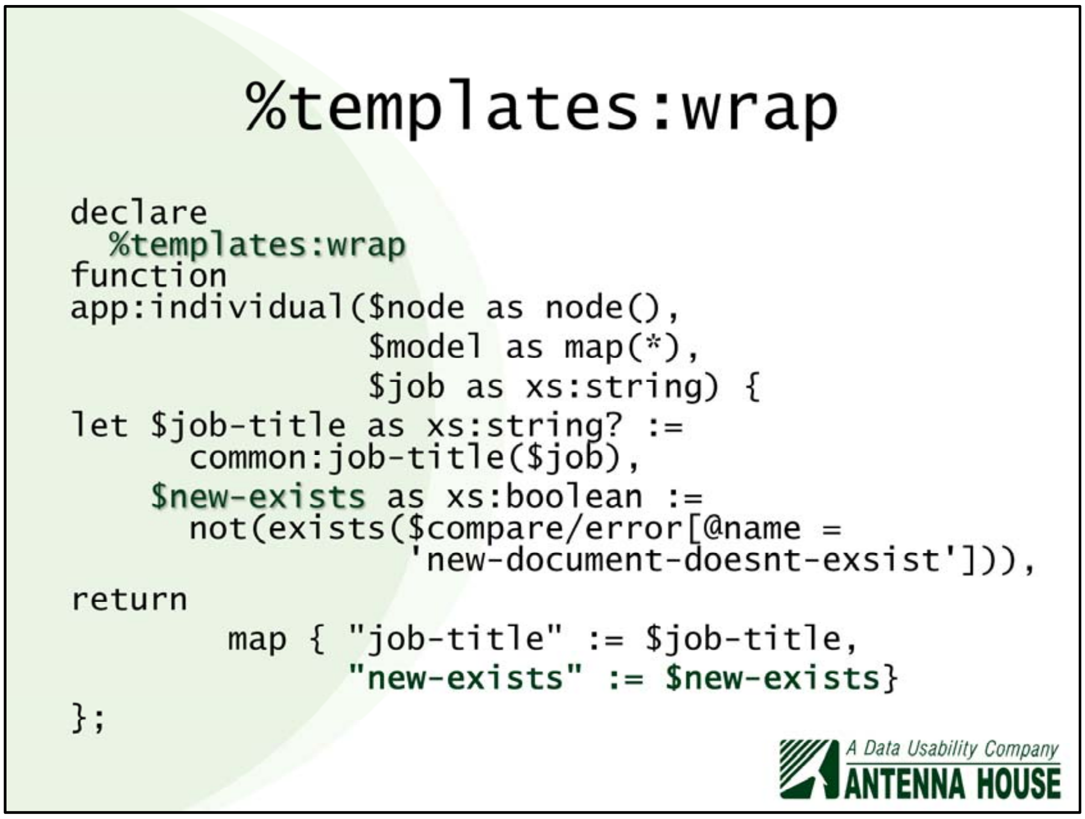
$node and $model

- $node – Current node from HTML
- $model – XQuery map of application data
- Additional function parameters from HTTP request, HTML session, or static defaults in XQuery

'$node' is the current node from the HTML.

'$model' is a map of application-specific data.

The templating mechanism can also populate values for function parameters that correspond to HTTP request parameters.

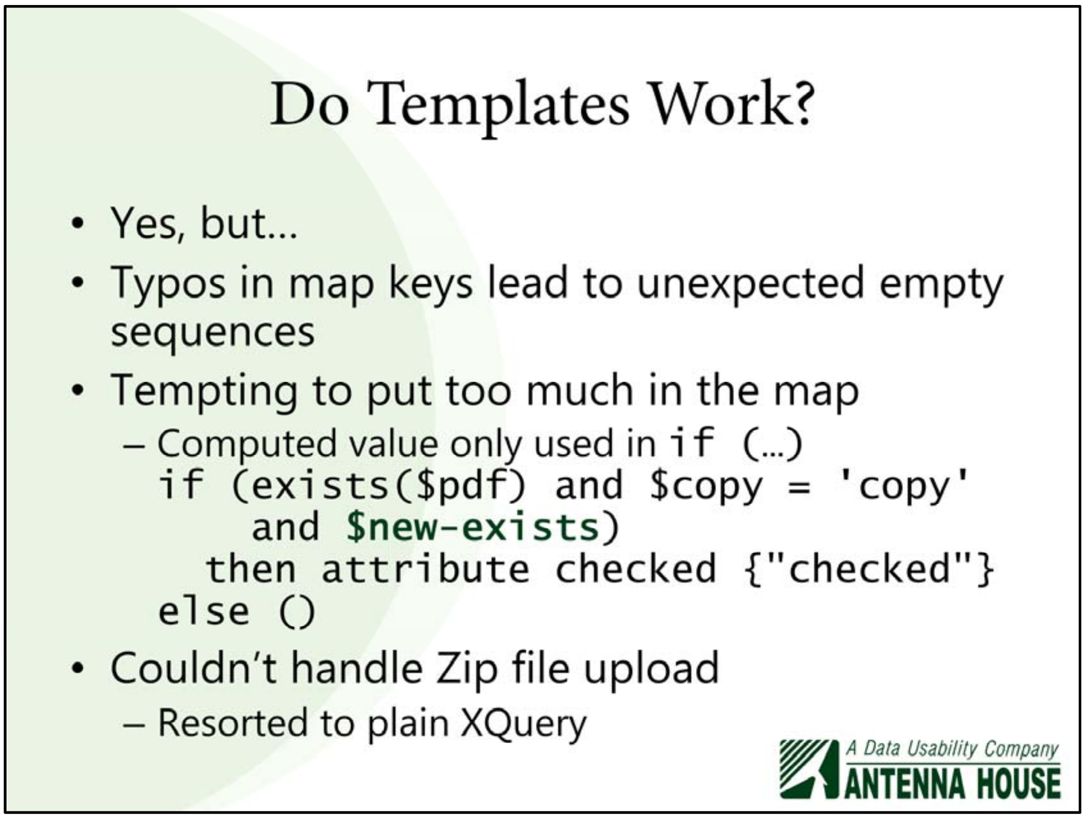


%templates:wrap

```
declare
  %templates:wrap
function
app:individual($node as node(),
               $model as map(*),
               $job as xs:string) {
let $job-title as xs:string? :=
      common:job-title($job),
    $new-exists as xs:boolean :=
      not(exists($compare/error[@name =
                  'new-document-doesnt-exsist'])),
return
        map { "job-title" := $job-title,
              "new-exists" := $new-exists}
};
```

A Data Usability Company
ANTENNA HOUSE

The alternative to a function returning HTML markup is that the function can, instead, modify the map of application-specific data. The modified map is available to the functions that are run from the HTML that the HTML for this function wraps.



Do Templates Work?

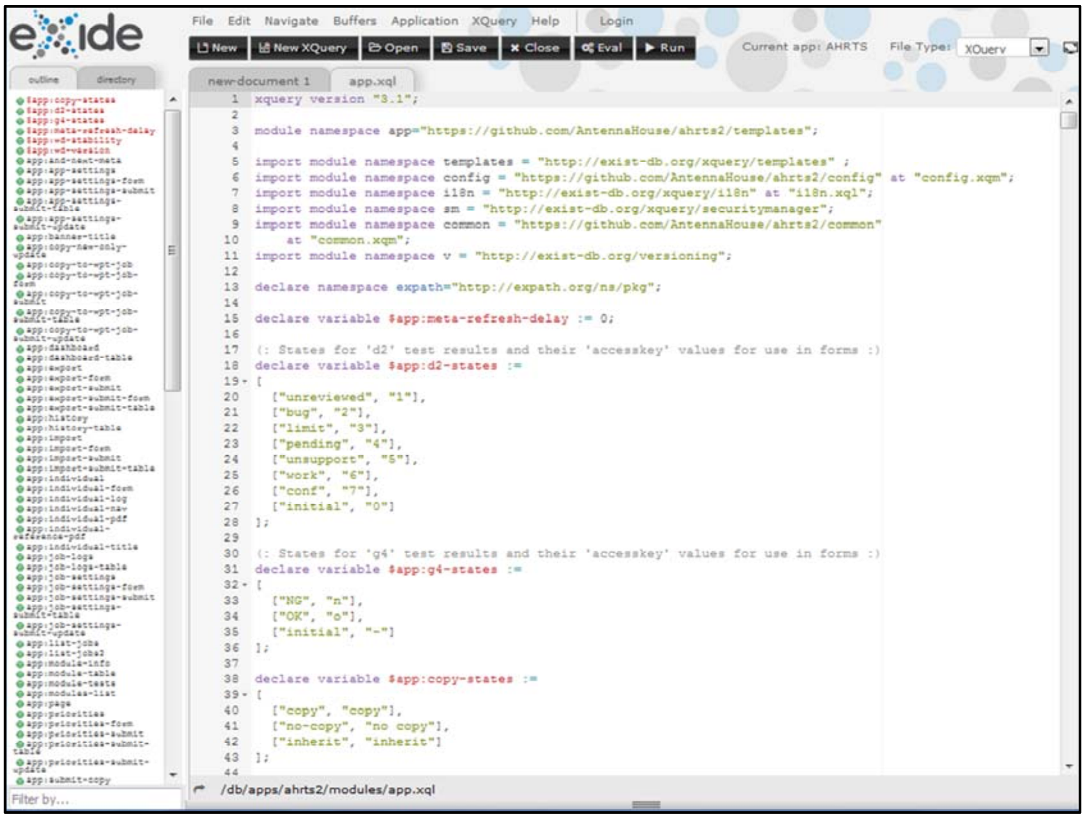
- Yes, but…
- Typos in map keys lead to unexpected empty sequences
- Tempting to put too much in the map
  - Computed value only used in `if (…)`
    ```
    if (exists($pdf) and $copy = 'copy'
          and $new-exists)
        then attribute checked {"checked"}
    else ()
    ```
- Couldn't handle Zip file upload
  - Resorted to plain XQuery

A Data Usability Company
ANTENNA HOUSE

Templates are useful. They make it easy to separate the HTML from the XQuery code.

However, it can be tempting to put too much into the map. The '$new-exists' that was added to the map by the code in the previous slide was actually only used in an 'if' statement. In the code shown here, it's possible that '$new-exists' is never used, despite being evaluated every time that the page is accessed.

Also, I could not get the templating mechanism to work with the page that uploads a Zip file, so that page is implemented using just XQuery.



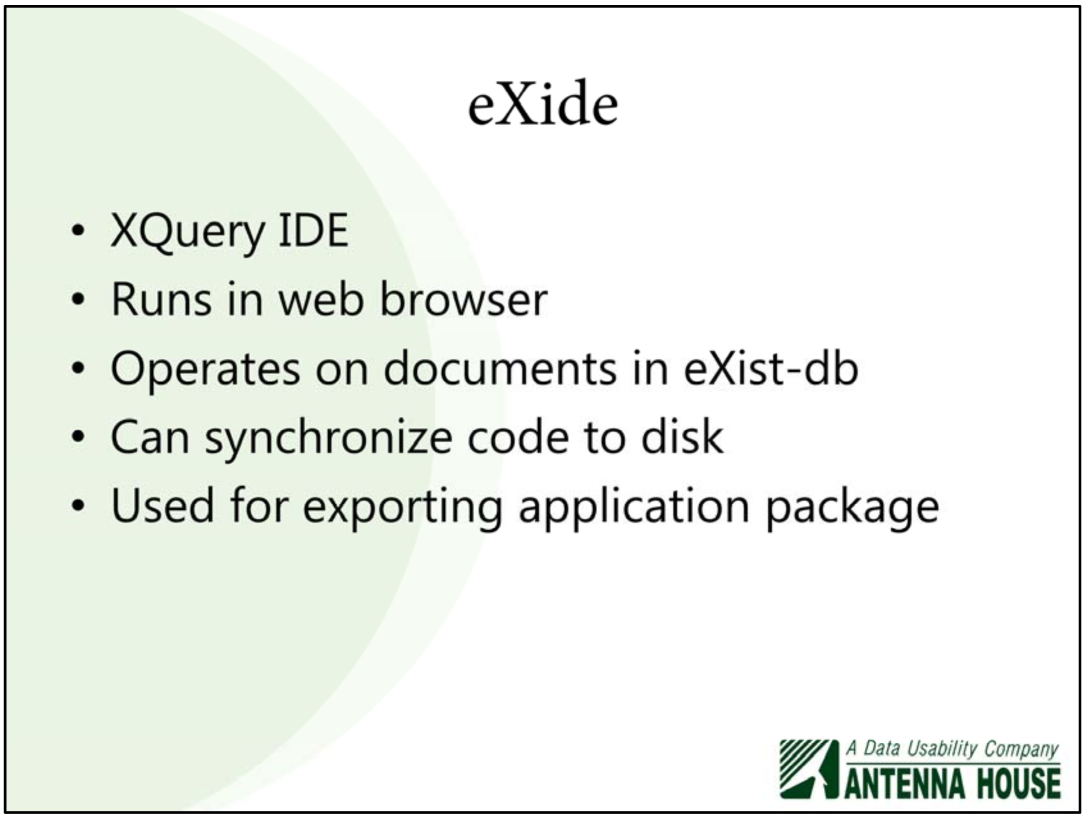
```
$app:copy-states
$app:d2-states
$app:g4-states
$app:meta-refresh-delay
$app:wd-stability
$app:wd-version
app:and-next-meta
app:app-settings
app:app-settings-form
app:app-settings-submit
app:app-settings-submit-table
app:app-settings-submit-update
app:banner-title
app:copy-new-only-update
app:copy-to-wpt-job
app:copy-to-wpt-job-form
app:copy-to-wpt-job-submit
app:copy-to-wpt-job-submit-table
app:copy-to-wpt-job-submit-update
app:dashboard
app:dashboard-table
app:export
app:export-form
app:export-submit
app:export-submit-form
app:export-submit-table
app:history
app:history-table
app:import
app:import-form
app:import-submit
app:import-submit-table
app:individual
app:individual-form
app:individual-log
app:individual-nav
app:individual-pdf
app:individual-reference-pdf
app:individual-title
app:job-logs
app:job-logs-table
app:job-settings
app:job-settings-form
app:job-settings-submit
app:job-settings-submit-table
app:job-settings-submit-update
app:list-jobs
app:list-jobs2
app:module-info
app:module-table
app:module-tests
app:modules-list
app:page
app:priorities
app:priorities-form
app:priorities-submit
app:priorities-submit-table
app:priorities-submit-update
app:submit-copy
```

```
xquery version "3.1";

module namespace app="https://github.com/AntennaHouse/ahrts2/templates";

import module namespace templates = "http://exist-db.org/xquery/templates" ;
import module namespace config = "https://github.com/AntennaHouse/ahrts2/config" at "config.xqm";
import module namespace i18n = "http://exist-db.org/xquery/i18n" at "i18n.xql";
import module namespace sm = "http://exist-db.org/xquery/securitymanager";
import module namespace common = "https://github.com/AntennaHouse/ahrts2/common"
    at "common.xqm";
import module namespace v = "http://exist-db.org/versioning";

declare namespace expath="http://expath.org/ns/pkg";

declare variable $app:meta-refresh-delay := 0;

(: States for 'd2' test results and their 'accesskey' values for use in forms :)
declare variable $app:d2-states :=
[
  ["unreviewed", "1"],
  ["bug", "2"],
  ["limit", "3"],
  ["pending", "4"],
  ["unsupport", "5"],
  ["work", "6"],
  ["conf", "7"],
  ["initial", "0"]
];

(: States for 'g4' test results and their 'accesskey' values for use in forms :)
declare variable $app:g4-states :=
[
  ["NG", "n"],
  ["OK", "o"],
  ["initial", "-"]
];

declare variable $app:copy-states :=
[
  ["copy", "copy"],
  ["no-copy", "no copy"],
  ["inherit", "inherit"]
];
```

/db/apps/ahrts2/modules/app.xql

This is a screenshot of the eXide editor that is part of eXist-db.

It runs in your browser, and it is the main tool for writing XQuery code.

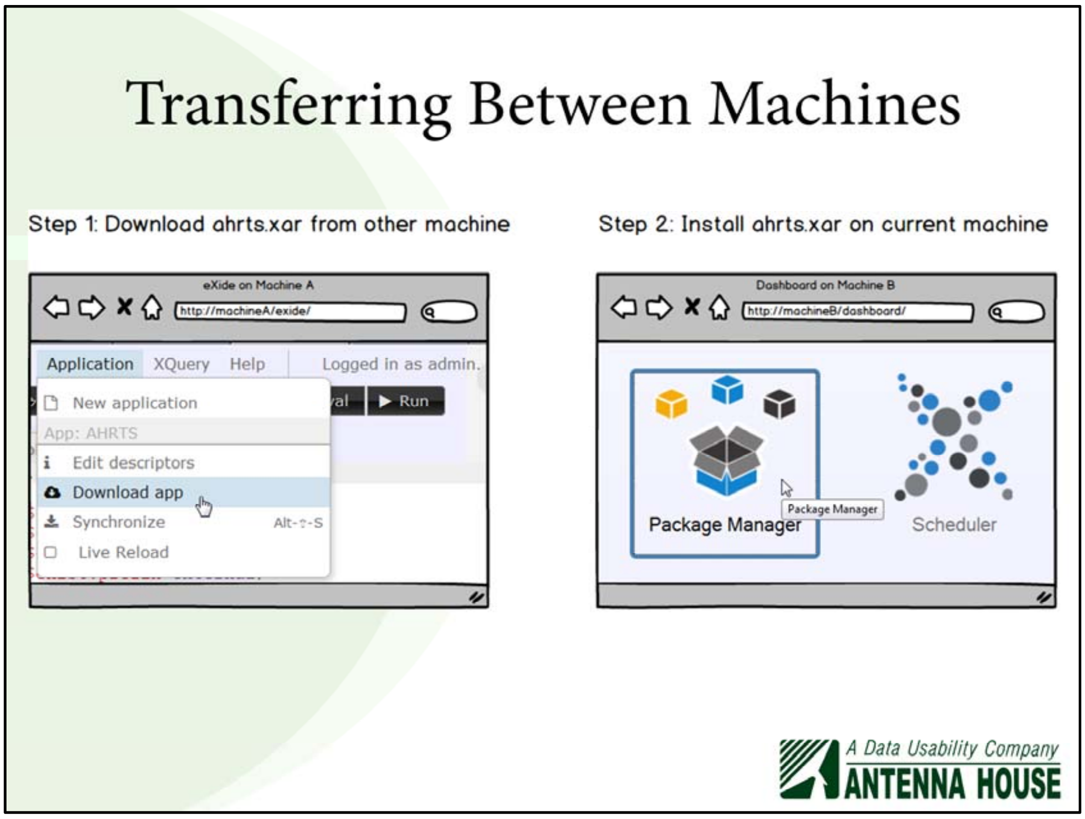


eXide

- XQuery IDE
- Runs in web browser
- Operates on documents in eXist-db
- Can synchronize code to disk
- Used for exporting application package

A Data Usability Company ANTENNA HOUSE

eXide can edit documents that are in the eXist-db database. Two of its other useful functions are synchronizing your XQuery code to disk (so that the code can be checked in to a version control system) and exporting application packages as '.xar' files.



Transferring Between Machines

Step 1: Download ahrts.xar from other machine

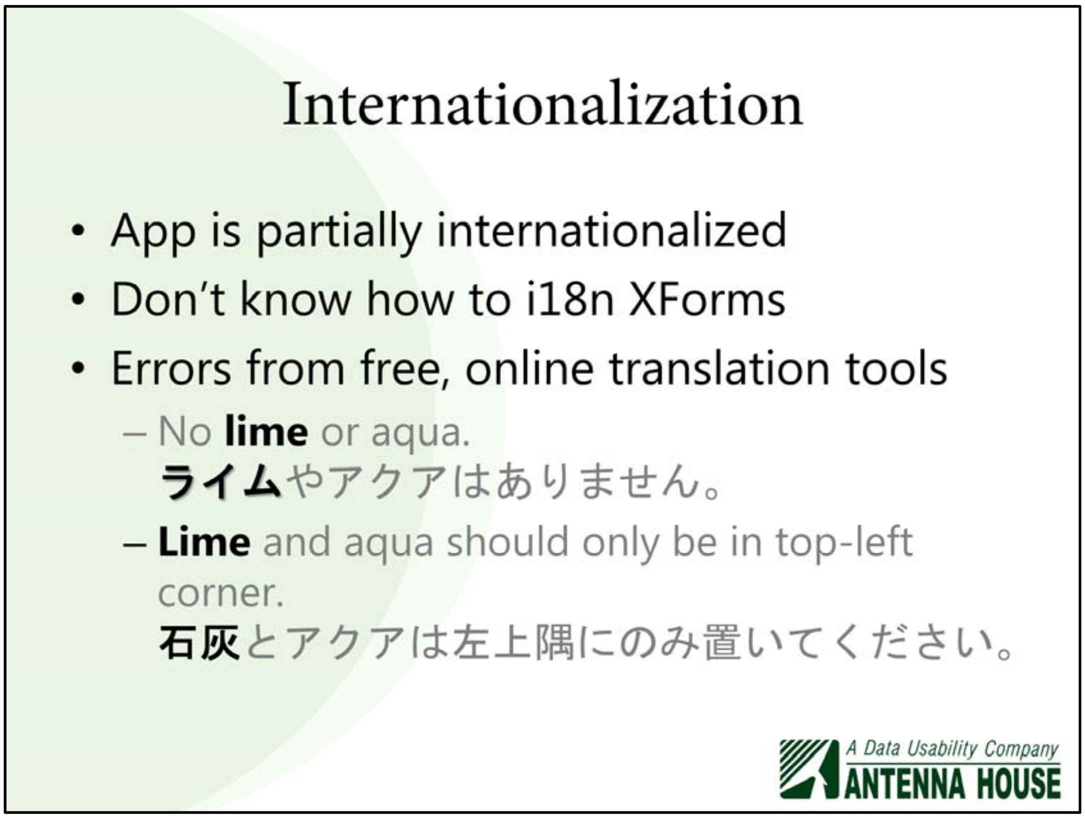
Step 2: Install ahrts.xar on current machine

Since eXide can synchronize your XQuery code to disk but cannot synchronize files in eXist-db to match files on the disk, it’s not straightforward to transfer XQuery code between machines.

In practice, I ended up exporting the app from one eXist-db instance and then installing the app on the eXist-db instance on the second machine.

If necessary, I could edit the XQuery files on the second machine and synchronize those files to disk on the second machine (and check the changed files into the distributed version control system), but getting the changed files back onto the first eXist-db instance required exporting the updated app from the second instance and installing that on the first instance.

This wasn’t needed very often, otherwise I would have found a way to update the XQuery code from a version control system check-out, possibly using either the eXist-db Ant task or the eXist-db Java client.



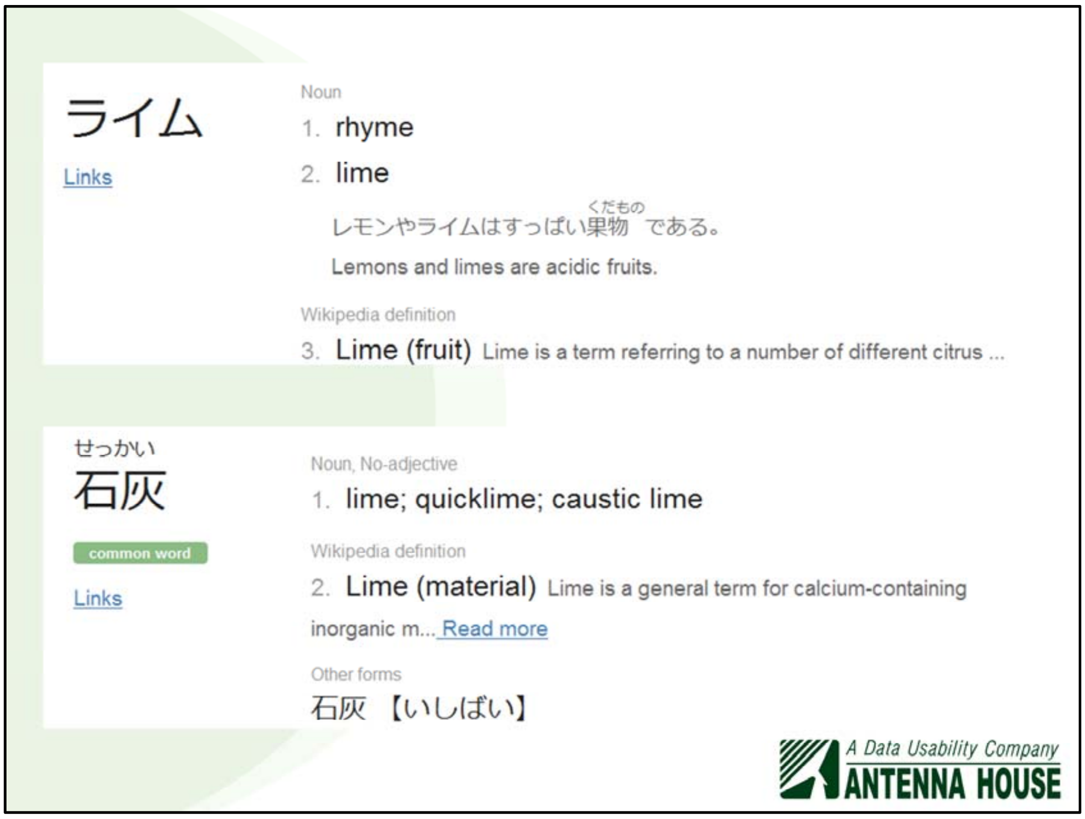
Internationalization

- App is partially internationalized
- Don’t know how to i18n XForms
- Errors from free, online translation tools
  - No **lime** or aqua.
    ライムやアクアはありません。
  - **Lime** and aqua should only be in top-left corner.
    石灰とアクアは左上隅にのみ置いてください。

A Data Usability Company
ANTENNA HOUSE

The app is currently partially internationalized. It is only partial both because I could not work out how to localize the XForms controls and because I was asked to undo some of my translations.

This example illustrates the problem with plucking translations from an online translation tool. These messages are from test result comments. Can you see the problem?



ライム

Links

Noun

1. rhyme
2. lime

レモンやライムはすっぱい果物（くだもの）である。

Lemons and limes are acidic fruits.

Wikipedia definition

3. Lime (fruit) Lime is a term referring to a number of different citrus ...

せっかい
石灰

common word

Links

Noun, No-adjective

1. lime; quicklime; caustic lime

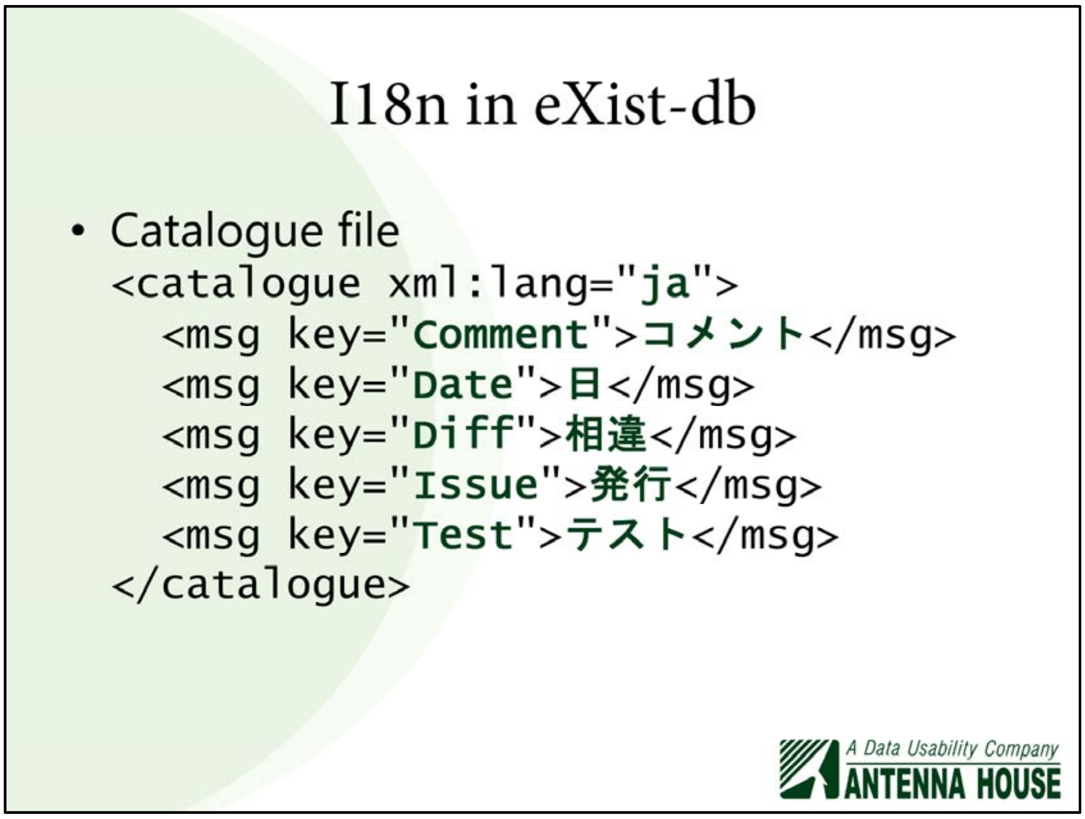
Wikipedia definition

2. Lime (material) Lime is a general term for calcium-containing inorganic m... Read more

Other forms

石灰 【いしばい】

This should make it clear to you.



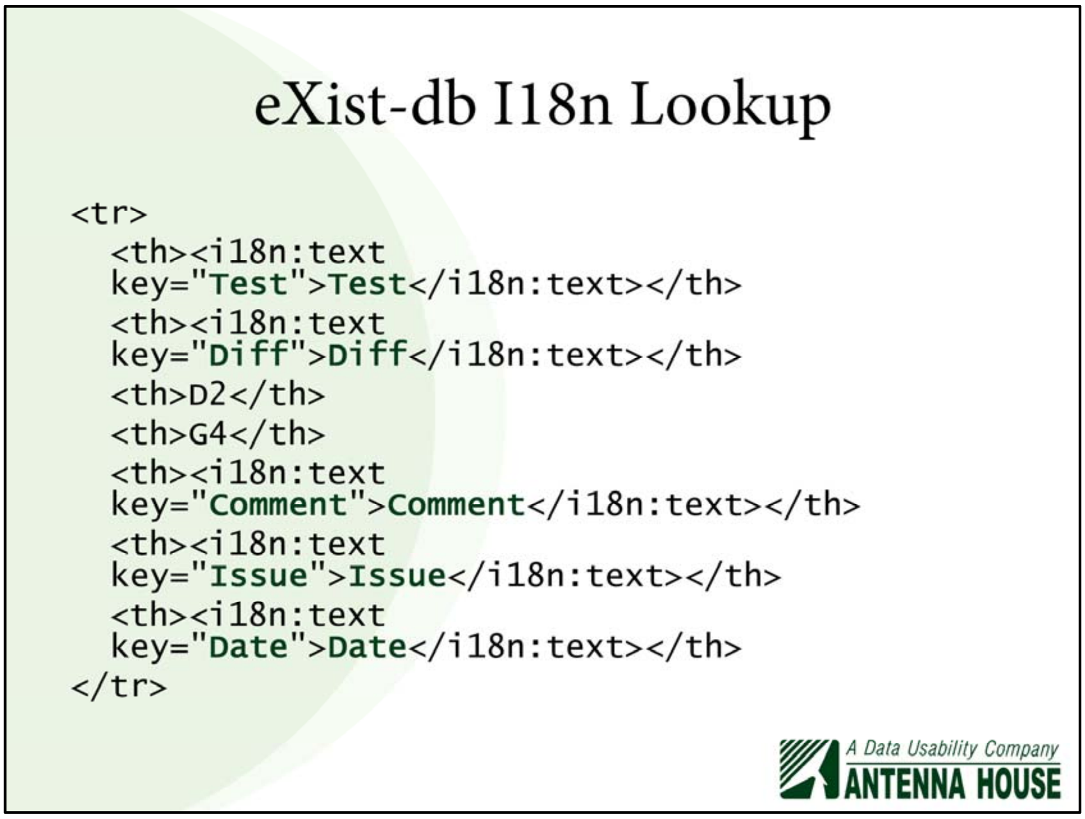
I18n in eXist-db

- Catalogue file

```
<catalogue xml:lang="ja">
  <msg key="Comment">コメント</msg>
  <msg key="Date">日</msg>
  <msg key="Diff">相違</msg>
  <msg key="Issue">発行</msg>
  <msg key="Test">テスト</msg>
</catalogue>
```

A Data Usability Company
ANTENNA HOUSE

The i18n library in eXist-db uses a catalogue file format that is structurally identical to Java XML property files.

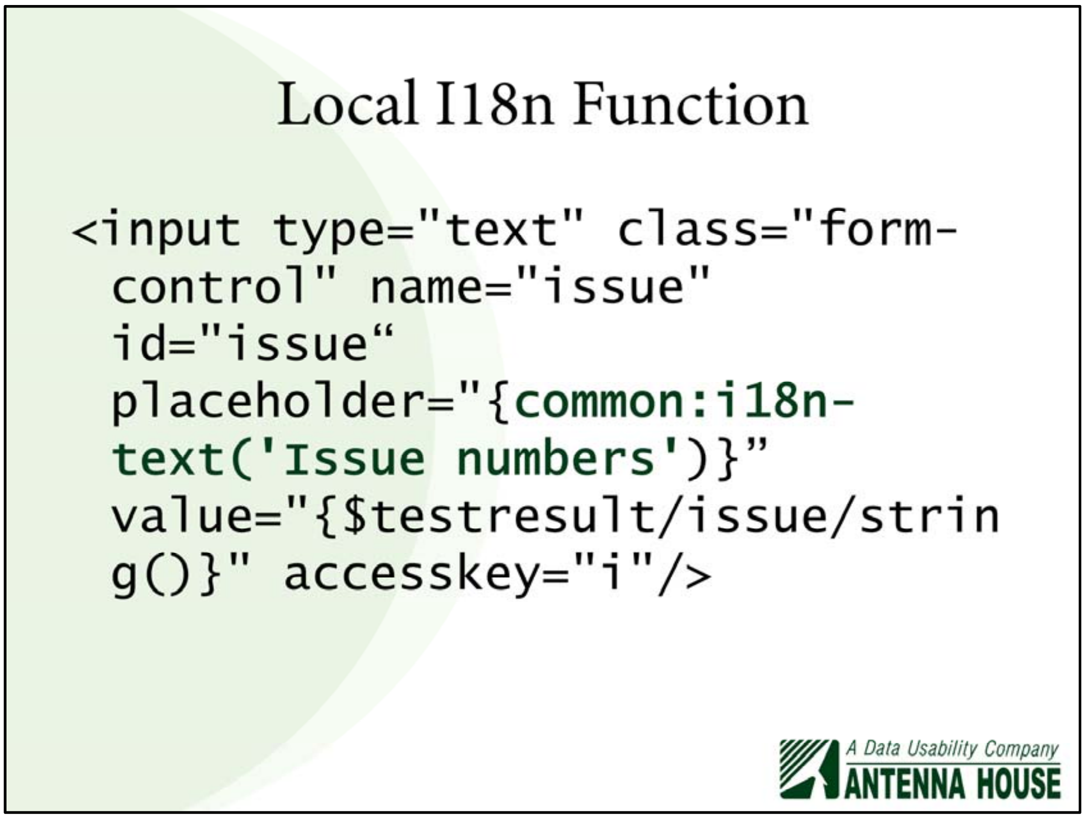


eXist-db I18n Lookup

```
<tr>
  <th><i18n:text
  key="Test">Test</i18n:text></th>
  <th><i18n:text
  key="Diff">Diff</i18n:text></th>
  <th>D2</th>
  <th>G4</th>
  <th><i18n:text
  key="Comment">Comment</i18n:text></th>
  <th><i18n:text
  key="Issue">Issue</i18n:text></th>
  <th><i18n:text
  key="Date">Date</i18n:text></th>
</tr>
```

A Data Usability Company
ANTENNA HOUSE

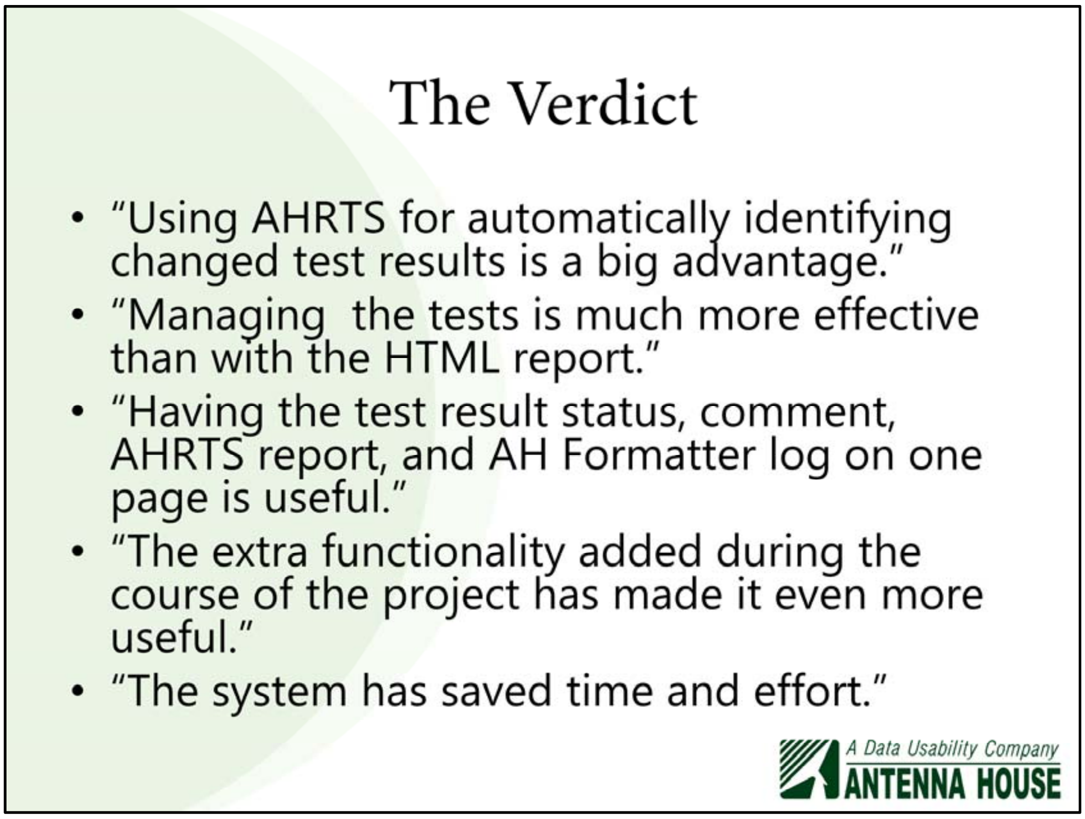
The i18n library replaces the content of the <i18n:text> elements with the localized text.



Local I18n Function

```
<input type="text" class="form-
  control" name="issue"
  id="issue“
  placeholder="{common:i18n-
  text('Issue numbers')}”
  value="{$testresult/issue/strin
  g()}" accesskey="i"/>
```

The <i18n:text> element isn’t useful for localizing attribute values, and I didn’t like repeating the text in the ‘key’ attributes, so I wrote my own i18n library.

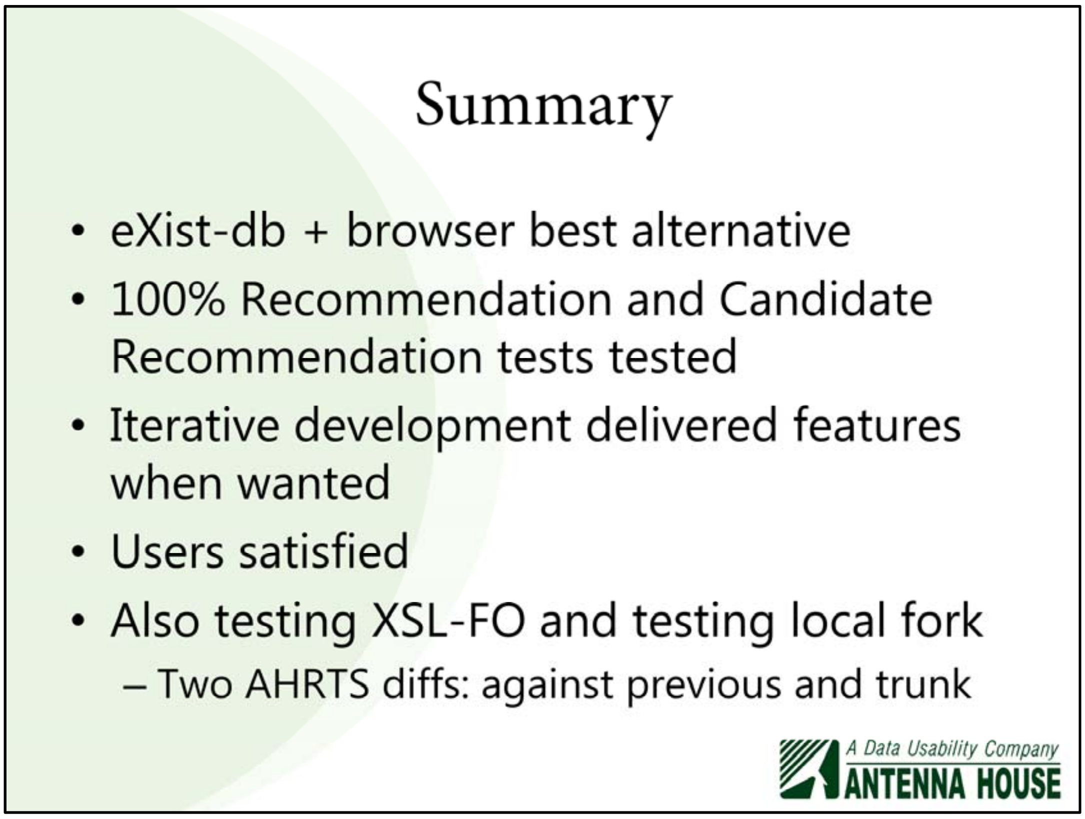


The Verdict

- “Using AHRTS for automatically identifying changed test results is a big advantage.”
- “Managing the tests is much more effective than with the HTML report.”
- “Having the test result status, comment, AHRTS report, and AH Formatter log on one page is useful.”
- “The extra functionality added during the course of the project has made it even more useful.”
- “The system has saved time and effort.”

A Data Usability Company ANTENNA HOUSE

The verdict from my colleagues in Japan was uniformly positive.



Summary

- eXist-db + browser best alternative
- 100% Recommendation and Candidate Recommendation tests tested
- Iterative development delivered features when wanted
- Users satisfied
- Also testing XSL-FO and testing local fork
  - Two AHRTS diffs: against previous and trunk

A Data Usability Company ANTENNA HOUSE

In summary:

• Using eXist-db and putting the test results in the browser was the best alternative that I tried
• 100% of the tests for CSS modules at the Recommendation and Candidate Recommendation stages have been tested
• The iterative development delivered features as they were wanted, including features (such as listing results for the entire test suite) that were removed and reinstated
• The users are satisfied with the result
• My testing has gone full circle: after basing the CSS testing on how I tested XSL-FO, I now present XSL-FO test results based on how I present CSS test results
• Another variant for checking results from a local fork of the AH Formatter code shows two AHRTS individual reports for each test: one with the differences between the current and previous results for the forked code, and the other between the current results for the forked code and the current results for the trunk code

A Data Usability Company
ANTENNA HOUSE

So Long, and Thanks for All the Fish

Tony Graham
XML Division
Antenna House, Inc.
tgraham@antenna.co.jp
tony@antennahouse.com
@tgraham_antenna

So long, and thanks for all the fish.