Edd Sowden
Senior Developer
Government Digital Service
@edds
Accessibility and how to get the most from your screenreader

What even is a table? A quick look at Accessibility APIs
I’m from the Government
We’re a team at the heart of government building digital public services
We started by building GOV.UK
The best place to find government services and information
Apply for, renew or update a UK passport online

You can apply for, renew or update your passport and pay for it online. You’ll have to print out a form at the end.

You must sign and date the form, add any documents or photographs that are needed, and return it for processing.

Before you start Other ways to apply

You’ll need a debit or credit card to use this service.

It should take 6 weeks to get your first adult passport once your form has been received. For all other application types, it should take 3 weeks.

It can take longer if more information is needed or your application hasn’t been filled out correctly.
Running a limited company

1. Directors' responsibilities

As a director of a limited company, the law says you must:

- try to make the company a success, using your skills, experience and judgment
- follow the company’s rules, shown in its articles of association
- make decisions for the benefit of the company, not yourself
- tell other shareholders if you might personally benefit from a transaction the company makes
- keep company records and report changes to Companies House and HM Revenue and Customs (HMRC)
- make sure the company’s accounts are a ‘true and fair view’ of the business’ finances

2. Taking money out of a limited company

3. Company changes you must report

4. Company and accounting records

5. Company annual return

6. Signs, stationery and promotional material
Guidance

Lumpy skin disease: how to spot and report the disease

From: Department for Environment, Food & Rural Affairs and Animal and Plant Health Agency
First published: 26 August 2014
Last updated: 1 October 2014, see all updates
Part of: Notifiable diseases in animals
Applies to: England, Scotland and Wales

How to spot lumpy skin disease, what to do if you suspect it and measures to prevent its spread.

Contents

How to spot lumpy skin disease
How lumpy skin disease is spread

Lumpy skin disease affects cattle and water buffalo. Humans aren’t affected.

It has never been present in Great Britain.
Apply for or renew a Blue Badge

Answer questions about your disability, check your eligibility and fill in an application form to apply for or renew a Blue Badge. Your application will then be sent to your council for a decision.

Start now on the Blue Badge eligibility tool

Before you start

You’ll need the following to apply for or renew your Blue Badge

- the details of your current Blue Badge (if you have one)
- a digital or signed photo
- your National Insurance number
- proof of identification
- proof of residency

Ask your council to find out how long before the expiry date you
Financial help if you're disabled

1. Overview
   There is a wide range of disability-related financial support, including benefits, tax credits, payments, grants and concessions.

   The main disability and sickness benefits are:
   - Disability Living Allowance or Personal Independence Payment
   - Attendance Allowance
   - Employment and Support Allowance

   Depending on your circumstances, you might also be able to get:
   - Industrial Injuries Benefit if you’re disabled as a result of work
   - Constant Attendance Allowance if you need daily care and attention

2. Disability and sickness benefits
3. Vehicles and transport
4. Home and housing
5. On a low salary
6. Television licence discount
7. VAT relief for disabled people
8. Work related injuries or illness
9. If you've served in Her Majesty's Armed Forces

Carers and disability benefits

- Benefits calculators
- Carer's Allowance
- More

Elsewhere on GOV.UK

- Apply for a Council Tax discount
- Apply for equipment for your home if you're disabled
- Disability day care centres
- Get meals at home (meals on wheels)
- Looking for work if you're disabled
GOV.UK should work for everyone
I wrote a blog post
Google Analytics is widely used to gather information about how users interact with websites. As a developer, designer or product owner it’s really useful to use the information it collects to inform decisions. However, I often find that the Google Analytics interface can get in the way of getting the right data out quickly.

Instead I prefer to use the Google Analytics Query Explorer to get information out. The Query Explorer was initially designed to help developers construct queries for the Google Analytics API. It also works really well for getting specific information out of Google Analytics when you want to do your own data analysis.

The way that the Google Analytics API works is very similar to using an OLAP cube, you define the dimensions and metrics for the data you want to retrieve. You can think of dimensions as the discrete data sets, for example: browser name, page path, custom event name. Metrics on the other hand are the measurable value to go with your dimensions, for example: pageviews, page load time, goal completion numbers.

Using dimensions and metrics along with some filters and sorting I find you can easily get a much wider array of information out than you would be able to using the Google Analytics web interface.

For example if you wanted to see what browsers your users are using you could run:

```
dimensions = ga:browser
metrics    = ga:users
sort       = -ga:users
```

Using a GOV.UK analytics profile this would get you something like: (view with your own data in the Query Explorer)

<table>
<thead>
<tr>
<th>Browser</th>
<th>Hour</th>
<th>Day of Week</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>0</td>
<td>5895</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>1</td>
<td>6814</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>2</td>
<td>8115</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>3</td>
<td>8000</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>4</td>
<td>8077</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Then using the TSV export button in the Query Explorer, found at the bottom of the page, I can import the results into a Google Sheets and produce a quick graph of the data.

For another example if we had a page which used a query string to let users filter by a date field. You could get back all the different dates people
Which would get you something like: 
(view with your own data in the
Query Explorer)

<table>
<thead>
<tr>
<th>Browser</th>
<th>Hour</th>
<th>Day of Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>2</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>3</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Then using the TSV export button in the Query Explorer, found at the bottom of the page, I can import the data into a Google Sheet and...
```css
table {
  display: block;
  overflow-x: scroll;
}
```
Which would get you something like: *(view with your own data in the Query Explorer)*

<table>
<thead>
<tr>
<th>Browser</th>
<th>Hour</th>
<th>Day of We</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>2</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>3</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>4</td>
</tr>
</tbody>
</table>

Then using the TSV export button in the Query Explorer, found at the bottom of the page, I can import the results into a Google Sheets and
What does this do to accessibility software?
Listen to your websites
How many of you use Apple products regularly?
How many of you test your websites in a screenreader?
Voiceover comes with OS X and iOS
The Voiceover Training is really good
Let’s listen to it before I made the change
<table>
<thead>
<tr>
<th>Browser</th>
<th>Hour</th>
<th>Day of Week</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>0</td>
<td>5895</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>1</td>
<td>6814</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>2</td>
<td>8115</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>3</td>
<td>8000</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>4</td>
<td>8077</td>
</tr>
</tbody>
</table>

...
And once I had added the extra CSS...
<table>
<thead>
<tr>
<th>Browser</th>
<th>Hour</th>
<th>Day of Week</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>0</td>
<td>5895</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>1</td>
<td>6814</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>2</td>
<td>8115</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>3</td>
<td>8000</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>00</td>
<td>4</td>
<td>8077</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The CSS display property affects how accessibility tech interprets the element
<div class="table-wrapper">
  <table> ... </table>
</div>
I had lunch
“Can’t you just set an aria role of table?”
@jaffathecake
<table role="table">
Minimum viable table:

<table>
<tr><td>Cell</td></tr>
</table>
<!DOCTYPE html>
<html>
  <head>...
  </head>
  <body>
    <table>
      <tbody>
        <tr>
          <td>cell</td>
        </tr>
      </tbody>
    </table>
</body>
</html>

Node has no text alternative.

Accessibility node not exposed

Element not interesting for accessibility.
chrome://flags/#enable-devtools-experiments
Enable Developer Tools experiments. Mac, Windows, Linux, Chrome OS
Enable Developer Tools experiments. Use Settings panel in Developer Tools to toggle individual experiments. #enable-devtools-experiments
Enable
Settings

General
- Disable cache (while DevTools is open)
- Disable JavaScript

Appearance
- Don’t show emulation warnings
- Enable Cmd + 1-9 shortcuts to switch panels
- Don’t show Chrome Data Saver warning
- Disable paused state overlay

Elements
- Color format: As authored
- Show user agent shadow DOM
- Word wrap
- Show rulers

Sources
- Link handling: auto

Network
- Color-code resource types

Profiler
- Hide chrome frame in Layers view
- Show advanced heap snapshot properties
- Record heap allocation stack traces
- High resolution CPU profiling

Console
- Hide network messages
- Log XMLHttpRequests
- Preserve log upon navigation
- Show timestamps

Extensions
- Link handling: auto
Experiments

WARNING: These experiments could be dangerous and may require restart.

- Accessibility Inspection
- Animation Inspection
- Allow custom UI themes
- Empty sourcemap auto-stepping
- FileSystem inspection
- Layers panel
- Material design
- Private script inspection
- Promise inspector
- Security panel
- Step into async
Node has no text alternative.

Accessibility Node
Accessibility node not exposed
Element not interesting for accessibility.
So our minimum viable table isn’t a table?
Let’s have a quick recap on Accessibility APIs
Current date is Tue 1-01-1980
Enter new date:
Current time is 7:48:27.13
Enter new time:

The IBM Personal Computer DOS
Version 1.10 (C)Copyright IBM Corp 1981, 1982

A>dir/w
COMMAND COM FORMAT COM CHKDSK COM SYS COM DISKCOPY COM
DISKCOMP COM COMP COM EXE2BIN EXE MODE COM EDLIN COM
DEBUG COM LINK EXE BASIC COM BASICA COM ART BAS
SAMPLES BAS MORTGAGE BAS COLORBAR BAS CALENDAR BAS MUSIC BAS
DONKEY BAS CIRCLE BAS PIECHART BAS SPACE BAS BALL BAS
COMM BAS
26 File(s)
A>dir command.com
COMMAND COM 4959 5-07-82 12:00p
1 File(s)
A>
Microsoft Windows
MS-DOS Executive

Version 1.01
Copyright © 1985, Microsoft Corp.

Disk Space Free: 30024K
Memory Free: 303K
Accessibility APIs

Windows 7: MSAA/IAccessible
OSX: NSAccessibility
iOS: UI Accessibility
Android: Accessibility Framework
role
name
state
children
HTML to Platform Accessibility APIs Implementation Guide

W3C Working Draft 01 October 2013

This version:  
http://www.w3.org/TR/2013/WD-html-aapi-20131001/

Latest published version:  
http://www.w3.org/TR/html-aapi/

Latest editor's draft:  

Previous version:  
http://www.w3.org/TR/2012/WD-html-aapi-20130901/

Editors:  
Steve Faulkner, The Paciello Group, sfaulkner@paciellogroup.com  
Cynthia Shelly, Microsoft, cyns@microsoft.com  
Jason Kiss, New Zealand Government, jason.kiss@dia.govt.nz  
Alexander Surkov, Mozilla Foundation, asurkov@mozilla.com

Copyright © 2011-2013 W3C® (MIT, ERCIM, Keio, Beihang), All Rights Reserved. W3C liability, trademark and document use rules apply.
### Role Mapping Matrix

<table>
<thead>
<tr>
<th></th>
<th>HTML4</th>
<th>HTML5</th>
<th>WAI-ARIA</th>
<th>MSAA + UIA Express</th>
<th>MSAA + IAccessible2</th>
<th>AT-SPI</th>
<th>AX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>none</td>
<td>Role: ROLE_SYSTEM_TABLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Use MSAA or UIA guidance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Role: ROLE_SYSTEM_TABLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Relations: IA2_RELATION_LABELLED_BY by child caption if any</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Interface: IAccessibleTable2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Control Type: Data Grid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Control Pattern: Table</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Role: ROLE_TABLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Relations: RELATION_LABELLED_BY by child caption if any</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Interface: Table</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AXRole: AXTable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AXSubrole: (nil)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AXRoleDescription: table</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Role: ROLE_TABLE**

- Relations: RELATION_LABELLED_BY by child caption if any

**Interface: Table**

**AXRole: AXTable**

**AXSubrole: (nil)**

**AXRoleDescription: table**
Browsers use the DOM and CSS to generate an accessibility tree. This is then passed onto accessibility APIs.
So what does it look like for a “real” table?
What do other browsers think of our minimum viable table?
<title>JS Bin</title>
<body>
<table>
  <tbody>
    <tr>
      <td>Cell</td>
    </tr>
  </tbody>
</table>
<script type="text/javascript" async src="http://www.google-analytics.com/plugins/ua/linkid.js"></script>
<script async src="/www.google-analytics.com/analytics.js"></script>
<script src="/static.jsbin.com/js/render/edit.js?3.34.3"></script>
</body>
"Accessibility Inspector" would like to control this computer using accessibility features.

Grant access to this application in Security & Privacy preferences, located in System Preferences.

Open System Preferences  Deny
Security & Privacy

Privacy

Allow the apps below to control your computer.

- Accessibility Inspector
- Dropbox

Click the lock to prevent further changes.

Advanced...
JAWS
JAWS advances to the next table with the ’t’ shortcut
<table>
  <tr><td>Cell</td></tr>
</table>
Browsers have heuristics to protect us from table based layouts
<table>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  <tr>
    <td>...</td><td>...</td>
  </tr>
</table>
<table>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  <tr>
    <td>...</td><td>...</td>
  </tr>
</table>
<table>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  [ ... 18 more rows ... ]
</table>
<table>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  [ ... 18 more rows ... ]
</table>
<table>
<thead>
<tr>
<th>Green</th>
</tr>
</thead>
</table>

```html
<table summary="colours">
  <tr><td>Green</td></tr>
</table>
```
<table summary="colours">
  <tr><td>Green</td></tr>
</table>

![Browser logos with green checkmark and red X]
<table>
  <tr>
    <th>...</th><td>...</td>
  </tr>
  <tr>
    <th>...</th><td>...</td>
  </tr>
</table>
<table>
  <tr>
    <th>...</th><td>...</td>
  </tr>
  <tr>
    <th>...</th><td>...</td>
  </tr>
</table>
```html
<table { display: block; }

<table>
  <tr>
    <th>...</th><td>...</td>
  </tr>
  <tr>
    <th>...</th><td>...</td>
  </tr>
</table>
```
table { display: block; }

<table>
  <tr>
    <th>…</th><td>…</td>
  </tr>
  <tr>
    <th>…</th><td>…</td>
  </tr>
</table>
\begin{tabular}{|c|c|}
\hline
... & ...
\hline
... & ...
\hline
\end{tabular}
```html
td { border-bottom: 1px solid; }

<table>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  <tr>
    <td>...</td><td>...</td>
  </tr>
</table>
```
td { background: grey; }

<table>
  <tr>
    <td>...</td><td>...</td>
  </tr>
  <tr>
    <td>...</td><td>...</td>
  </tr>
</table>
td { background: grey; }

<table>
   <tr>
      <td>…</td><td>…</td>
   </tr>
   <tr>
      <td>…</td><td>…</td>
   </tr>
</table>
```html
<table border-collapse: collapse; }
td { background: grey; }

<table>
  <tr>
    <td>…</td><td>…</td>
  </tr>
  <tr>
    <td>…</td><td>…</td>
  </tr>
</table>
```
```html
table {  border-collapse: collapse; }  
td {  background: grey; }  

<table>
  <tr>
    <td>…</td><td>…</td>
  </tr>
  <tr>
    <td>…</td><td>…</td>
  </tr>
</table>```
Let’s have a quick look under the hood in Chrome
bool AXTable::isDataTable() const
{
    if (!m_layoutObject || !node())
        return false;

    // Do not consider it a data table if it has an ARIA role.
    if (hasARIARole())
        return false;

    // When a section of the document is contentEditable, all tables should be
    // treated as data tables, otherwise users may not be able to work with rich
    // text editors that allow creating and editing tables.
    if (node() && node()->hasEditableStyle())
        return true;

    // This employs a heuristic to determine if this table should appear.
    // Only "data" tables should be exposed as tables.
    // Unfortunately, there is no good way to determine the difference
    // between a "layout" table and a "data" table.
    LayoutTable* table = toLayoutTable(m_layoutObject);
    Node* tableNode = table->node();
    if (!isHTMLTableElement(tableNode))
        return false;

    // Do not consider it a data table if any of its descendants have an ARIA role.
    HTMLTableElement* tableElement = toHTMLTableElement(tableNode);
    if (elementHasAriaRole(tableElement->tHead()))
        return false;
    if (elementHasAriaRole(tableElement->tFoot()))
        return false;

   RefPtrWillBeRawPtr<HTMLCollection> bodies = tableElement->tBodies();
    for (unsigned bodyIndex = 0; bodyIndex < bodies->length(); ++bodyIndex) {
        Element* bodyElement = bodies->item(bodyIndex);
        // Do not consider it a data table if any of its descendants have an ARIA role.
        if (elementHasAriaRole(bodyElement->tHead()))
            return false;
        if (elementHasAriaRole(bodyElement->tFoot()))
            return false;
    }
bool AXTable::isDataTable() const
{
    if (!m_layoutObject || !node())
        return false;

    // Do not consider it a data table if it has an ARIA role.
    if (hasARIARole())
        return false;

    // When a section of the document is contentEditable, all tables should be
    // treated as data tables, otherwise users may not be able to work with rich
    // text editors that allow creating and editing tables.
    if (node() && node()->hasEditableStyle())
        return true;

    // This employs a heuristic to determine if this table should appear.
    // Only "data" tables should be exposed as tables.
    // Unfortunately, there is no good way to determine the difference
    // between a "layout" table and a "data" table.

    LayoutTable* table = toLayoutTable(m_layoutObject);
    Node* tableNode = table->node();
    if (!isHTMLTableElement(tableNode))
        return false;

    // Do not consider it a data table if any of its descendants have an ARIA role.
    HTMLTableElement* tableElement = toHTMLTableElement(tableNode);
    if (elementHasAriaRole(tableElement->tHead()))
        return false;
    if (elementHasAriaRole(tableElement->tFoot()))
        return false;

    RefPtrWillBeRawPtr<HTMLCollection> bodies = tableElement->tBodies();
    for (unsigned bodyIndex = 0; bodyIndex < bodies->length(); ++bodyIndex) {
        Element* bodyElement = bodies->item(bodyIndex);
// cells have borders, or use attributes like headers, abbr, scope or axis
167. table->recalcSectionsIfNeeded();
168. LayoutTableSection* firstBody = table->firstBody();
169. if (firstBody)
170. {
171.     return false;
172. }
173. int numCols = firstBody->numColumns();
174. int numRows = firstBody->numRows();
175. // If there's only one cell, it's not a good AXTable candidate.
176. if (numRows == 1 && numCols == 1)
177.     return false;
178. // If there are at least 20 rows, we'll call it a data table.
179. if (numRows >= 20)
180.     return true;
181. // Store the background color of the table to check against cell's background colors.
182. const ComputedStyle* tableStyle = table->style();
183. if (!tableStyle)
184.     return false;
185. Color tableBGCColor = tableStyle->visitedDependentColor(CSSPropertyBackgroundColor);
186. // check enough of the cells to find if the table matches our criteria
187. // Criteria:
188. // 1) must have at least one valid cell (and)
189. // 2) at least half of cells have borders (or)
190. // 3) at least half of cells have different bg colors than the table, and there is cell spacing
191. unsigned validCellCount = 0;
192. unsigned borderedCellCount = 0;
193. unsigned backgroundDifferenceCellCount = 0;
194. unsigned cellsWithTopBorder = 0;
195. unsigned cellsWithBottomBorder = 0;
196. unsigned cellsWithLeftBorder = 0;
197. unsigned cellsWithRightBorder = 0;
return true;

// if there is less than two valid cells, it's not a data table
if (validCellCount <= 1)
    return false;

// half of the cells had borders, it's a data table
unsigned neededCellCount = validCellCount / 2;
if (borderedCellCount >= neededCellCount
    || cellsWithTopBorder >= neededCellCount
    || cellsWithBottomBorder >= neededCellCount
    || cellsWithLeftBorder >= neededCellCount
    || cellsWithRightBorder >= neededCellCount)
    return true;

// half had different background colors, it's a data table
if (backgroundDifferenceCellCount >= neededCellCount)
    return true;

// Check if there is an alternating row background color indicating a zebra striped style pattern.
if (alternatingRowColorCount > 2) {

    Color firstColor = alternatingRowColors[0];
    for (int k = 1; k < alternatingRowColorCount; k++) {
        // If an odd row was the same color as the first row, it's not alternating.
        if (k % 2 == 1 && alternatingRowColors[k] == firstColor)
            return false;

        // If an even row is not the same as the first row, it's not alternating.
        if (!(k % 2) && alternatingRowColors[k] != firstColor)
            return false;
    }
    return true;
}

return false;
What about the massive fox in the room?
Firefox seems to be different depending on which screen reader you’re using.
// performance problems only.  Note, currently 'aAllowEmpty' flag is used for
// caption element only.  On another hand we create accessible object for
// the first entry of caption element (see
// HTMLTableAccessible::CacheChildren).
return !!elements->Item(1);

bool
HTMLTableAccessible::IsProbablyLayoutTable()
{
// Implement a heuristic to determine if table is most likely used for layout
// XXX do we want to look for rowspan or colspan, especialy that span all but a couple cells
// at the beginning or end of a row/col, and especially when they occur at the edge of a table?
// XXX expose this info via object attributes to AT-SPI

// XXX For now debugging descriptions are always on via SHOW_LAYOUT_HEURISTIC
// This will allow release trunk builds to be used by testers to refine the algorithm
// Change to #define SHOW_LAYOUT_HEURISTIC_DEBUG before final release

#ifdef SHOW_LAYOUT_HEURISTIC
#define RETURN_LAYOUT_ANSWER(isLayout, heuristic) \ 
{ \
  mLayoutHeuristic = isLayout ? \ 
    NS_LITERAL_STRING("layout table: ") heuristic) : \ 
    NS_LITERAL_STRING("data table: ") heuristic); \
return isLayout; \ 
}
#else
#define RETURN_LAYOUT_ANSWER(isLayout, heuristic) { return isLayout; }
#endif

DocAccessible* docAccessible = Document();
if (docAccessible) {
  uint64_t docState = docAccessible->State();
  if (docState & states::EDITABLE) { // Need to see all elements while document is being edited
    RETURN_LAYOUT_ANSWER(false, "In editable document");
  }
}

// Check to see if an ARIA role overrides the role from native markup, 
// but for which we still expose table semantics (treegrid, for example).
if (Role() != roles::TABLE)
  RETURN_LAYOUT_ANSWER(false, "Has role attribute");

if (mContent->HasAttr(kNameSpaceID_None, nsGkAtoms::role)) {
  // Role attribute is present, but overridden roles have already been dealt with.
  // Only landmarks and other roles that don't override the role from native 
  // markup are left to deal with here.
  RETURN_LAYOUT_ANSWER(false, "Has role attribute, weak role, and role is table");
bool HTMLTableAccessible::IsProbablyLayoutTable()
{
    // Implement a heuristic to determine if table is most likely used for layout
    // XXX do we want to look for rowspan or colspan, especially that span all but a couple cells
    // at the beginning or end of a row/col, and especially when they occur at the edge of a table?
    // XXX expose this info via object attributes to AT-SPI
    // XXX For now debugging descriptions are always on via SHOW_LAYOUT_HEURISTIC
    // This will allow release trunk builds to be used by testers to refine the algorithm
    // Change to #define SHOW_LAYOUT_HEURISTIC_DEBUG before final release
    #ifdef SHOW_LAYOUT_HEURISTIC
        #define RETURN_LAYOUT_ANSWER(isLayout, heuristic) \
        { \n            mLayoutHeuristic = isLayout ? \n                NS_LITERAL_STRING("layout table: " heuristic) : \n                NS_LITERAL_STRING("data table: " heuristic); \n            return isLayout; \n        }
    #else
        #define RETURN_LAYOUT_ANSWER(isLayout, heuristic) { return isLayout; }
    #endif

    DocAccessible* docAccessible = Document();
    if (docAccessible)
    {
        uint64_t docState = docAccessible->State();
        if (docState & nsGkAtoms::EDITABLE) { // Need to see all elements while document is being edited
            RETURN_LAYOUT_ANSWER(false, "In editable document");
        }
    }

    // Check to see if an ARIA role overrides the role from native markup,
    // but for which we still expose table semantics (treegrid, for example).
    if (Role() != roles::TABLE)
    RETURN_LAYOUT_ANSWER(false, "Has role attribute");

    if (mContent->HasAttr(kNameSpaceID_None, nsGkAtoms::role)) { // Role attribute is present, but overridden roles have already been dealt with.
        // Only landmarks and other roles that don't override the role from native
        // markup are left to deal with here.
        RETURN_LAYOUT_ANSWER(false, "Has role attribute, weak role, and role is table");
    }
}
if (childIdx > 0 && prevRowColor != rowColor)
    RETURN_LAYOUT_ANSWER(false, "2 styles of row background color, non-bordered");
}

// Check for many rows
const uint32_t kMaxLayoutRows = 20;
if (rowCount > kMaxLayoutRows) { // A ton of rows, this is probably for data
    RETURN_LAYOUT_ANSWER(false, ">= kMaxLayoutRows (20) and non-bordered");
}

// Check for very wide table.
nsIFrame* documentFrame = Document()->GetFrame();
nsSize documentSize = documentFrame->GetSize();
if (documentSize.width > 0) {
    nsSize tableSize = GetFrame()->GetSize();
    int32_t percentageOfDocWidth = (100 * tableSize.width) / documentSize.width;
    if (percentageOfDocWidth > 95) {
        // 3-4 columns, no borders, not a lot of rows, and 95% of the doc's width
        // Probably for layout
        RETURN_LAYOUT_ANSWER(true,
            "4 columns, table width is 95% of document width");
    }
}

// Two column rules
if (rowCount * colCount <= 10) {
    RETURN_LAYOUT_ANSWER(true, "2-4 columns, 10 cells or less, non-bordered");
}
if (HasDescendant(NS_LITERAL_STRING("embed")) ||
    HasDescendant(NS_LITERAL_STRING("object")) ||
    HasDescendant(NS_LITERAL_STRING("applet")) ||
    HasDescendant(NS_LITERAL_STRING("iframe"))) {
    RETURN_LAYOUT_ANSWER(true, "Has no borders, and has iframe, object, applet or iframe, typical of advertisements");
}
RETURN_LAYOUT_ANSWER(false, "no layout factor strong enough, so will guess data");

// HTMLCaptionAccessible

Relation HTMLCaptionAccessible::RelationByType(RelationType aType) {

}
// Check for very wide table.
nsIFrame* documentFrame = Document()->GetFrame();
nsSize documentSize = documentFrame->GetSize();
if (documentSize.width > 0) {
    nsSize tableSize = GetFrame()->GetSize();
    int32_t percentageOfDocWidth = (100 * tableSize.width) / documentSize.width;
    if (percentageOfDocWidth > 95) {
        // 3-4 columns, no borders, not a lot of rows, and 95% of the doc's width
        // Probably for layout
        RETURN_LAYOUT_ANSWER(true, "<= 4 columns, table width is 95% of document width");
    }
}

// Two column rules
if (rowCount * colCount <= 10) {
    RETURN_LAYOUT_ANSWER(true, "2-4 columns, 10 cells or less, non-bordered");
}

if (HasDescendant(NS_LITERAL_STRING("embed")) ||
    HasDescendant(NS_LITERAL_STRING("object")) ||
    HasDescendant(NS_LITERAL_STRING("applet")) ||
    HasDescendant(NS_LITERAL_STRING("iframe"))) {
    RETURN_LAYOUT_ANSWER(true, "Has no borders, and has iframe, object, applet or iframe, typical of advertisements");
}

RETURN_LAYOUT_ANSWER(false, "no layout factor strong enough, so will guess data");

// HTMLCaptionAccessible

 Relation
HTMLCaptionAccessible::RelationByType(RelationType aType)
{
    Relation rel = HyperTextAccessible::RelationByType(aType);
    if (aType == RelationType::LABEL_FOR)
        rel.AppendTarget(Parent());

    return rel;
}

role
HTMLCaptionAccessible::NativeRole();
So what does this all mean?
Accessibility is more than just markup
Listen to your websites
<div class="table-wrapper">
  <table> ... </table>
</div>
Thank you!

Special thanks to:
Alice Bartlet, Robin Whittleton, Tom Byers, Alex Muller, Léonie Watson, David Singleton, Gemma Leigh and Jake Archibald
Edd Sowden
Senior Developer
Government Digital Service
@edds