**I. ARIA grid role**

***Grid: aria-gridtype=”data/layout”***

*We need a property to distinguish data-bound content from grids used for layout, e.g. role=”grid” aria-gridtype=”data/layout” or denote the concept of layout grids to a entire new role.*

In Business Applications , a **row** in a data grid represents **objects in a collection**. Many functions are therefore **row-based because the row is the object (**object selection, object drill down etc.). The grid **columns** represent simple or complex potentially editable object **properties** typically **related** to the row object**.**

Layout grids, in contrast, represent just navigable entity containers for active content (tic-tac-toe-game). Every cell is by principle on an equal footing with the other. Cell content may be potentially editable (such as an array of checkboxes) but rows and columns do not play that object-related part, the **entire grid is the object**.

Consequently, in both grid variants the inner object organization is **different**. Assistive Technology MUST know the type differences to react appropriately for each type. Currently this is not possible without a subtype definition or a special different role for layout grids. When this is not 100% defined we put burden on AT cause heuristics to be developed there which will lead to differences across various vendors and increase chaos.

***Examples for data tables***

Data table with focusable rows (inner role is HTML table)

Data Grid

***Row: aria-rowtype=”(****string****)”***

*Row: the poor cousin in ARIA grid and table roles*

As stated above, row is NOT just a structural aid, it has ***meaning***. This goes deep. Entire row browsing in data grids is intuitively done by putting technically the keyboard focus on the row element. Since the row **represents** the element it just makes sense. But even if not so, indication is needed what special type this row may be very helpful for users of assistive technology.

Complex data tables may e.g. contain special rows containing sum or total info in grid cells. The mechanism using e.g. aria-label info on row for that is utterly buggy since this will overwrite potentially all text content info not mentioning that this use case is poorly defined and described. Using aria-describedby miserably fails also since rows are not considered as directly focusable and therefore describedby relations (for which some AT heuristics presume they are only to be evaluated for **active** elements) cannot be used either.

It I just a question of time when more and different elaborated row types of special meaning evolve, and ARIA should be prepared for that.

***Columnheader: active, aria-fixed=”true/false” and aria-filtered=”****(string)****”***

*We need more grid columnheader properties*

In complex data grids, sequence for some grid columns cannot be changed, for others it can. In consequence, on grid scrolling some columns remain at fixed position (like fixed rows in Excel). ARIA has no property to indicate that (e.g. for columnheader role). Since column auto scroll may appear during left/right cell keyboard navigation, it will be confusing if such fixed columns won’t be indicated.

Also, we have an aria-sort state but no aria-filtered info for a column. See example below.

Please also note that the sort and filter triggers are typically done as a function **of an active columnheader** that **is** also sort of a menu button – a thing ARIA is not covering since the implementation recommendation is based on the prerequisite the column header **contains** a menu button that directs the sorting.

***Keyboard navigation, object and object relation detection***

*The lookup algorithm for cell content detection and labelling + describing in ARIA grids is poorly documented by the ARIA group.*

While we have an extensive description about accessibleName mapping, tests with common screen readers reveal several things:

* When focusing a grid cell containing an active role, the role is **not always** recognized. This is valid for quite a few ARIA roles (spinbutton etc.)
* When focusing a grid cell containing a non-active role, the role is **never** recognized. (progressbar etc.)
* When focusing a grid cell containing an object of a given role, the object state and important properties is nearly **never** detected (no progressbar value, no invalid state for input, no ariaLabelledBy relations or describedby relations of inner object)

With other words, you lose the “**what is this”** and “**what status has it**” context.

In consequence, users have to **extra focus** the inner object to get all state and extended labelling information. This can be done by tabbing through the active elements bypassing the arrow key navigation. But doing so, another problem immediately pops up:

* When focusing an active object within a related grid cell, the cell relations (ariaLabelledBy, ariaDescribedBy to headers and descriptions) are **not detected** by AT. This is valid for **all** ARIA roles (input, button etc.).

With other words, you lose the “**where am I”** context here when you tab through the items in a row (like a list of input elements) which is a TYPICAL use case in data grids to enter data.

The reason for all this is that neither this expected behavior nor any DOM or Platform Accessibility Tree traversal algorithms have been published for AT vendors. Therefore, to implement this behavior, much extra JS glue code has to be written for complex data grids which is considered superfluous and affects performance.

**II. ARIA data input/output role need aria-valuestate / aria-valuestatetext properties**

*We need more than aria-invalid=”true/false”*

**aria-valuestate=”error/warning/info/custom”** and **aria-valuestatetext=”string”**

We need a property to signal that something is not OK with a value or it needs additional processing

**Examples**

Input, Checkbox... **Grid Rows (Object invalid state),** Listitems

Warning

Highlight

Error

