OUIA Documentation

Peter Savage, Ronny Pfannschmidt, Karel Hala

Oct 06, 2020

Contents

Welcome to the OUIA specification. The OUIA was designed to assist the various parties involed in web UI construction in creating an easy to use, frustration free experience for QE/QA testers. Its core concepts have been developed over many years and honed in real life experience. The goal is to impact developers as little as possible by enforcing standards upstream where possible.

The standard defines a number of responsibilities on:

- Component/Widget providers
- Page developers

CHAPTER 1

History

The OUIA specification project was created after years of work automating the ManageIQ project. The project had a very basic SOAP API that did not have feature parity with the UI. As such, despite the difficulty in creating such automation, in order to have a full regression suite of tests, the UI was the only option.

Several consistent themes emerged from the five years of efforts. These themes were distilled down into a wishlist. If only the UI could provide us with X. When the ManageIQ project started standardising on Patternfly, and with a new version of Red Hat Insights also using the Paternfly library, it felt like the right time to start looking at solidifying some of these requests and pushing them as upstream as we could.

Specifically we were looking for:

- Notification that any items had finished animating
- Notification that the page was considered safe to touch
- Web Components to declare themselves on the page

Though HTML elements can often be inferred, it is far easier to find them if there are explicit standard attributes to look for. The OUIA requests that framework component developers specifically include certain attributes to assist in the discovery of their elements on the page.

Developers of the site/application are required to adhere to certain other standards to assist in page function discovery, as well as to provide notification of the page being in an unsafe state.

CHAPTER 2

Open UI Automation (OUIA) 1.0-RC

2.1 Metadata

2.2 Abstract

This specification is designed to promote certain key guidelines to follow when creating a new web framework or application. Its goal is to ease the burden of individuals wishing to create and maintain automated testing environments.

The OUIA is composed of multiple sections. An element of an app/framework can be said to be compliant with either all of the specification or one or more of its parts. As such it is defined as a *composable* specification. The reason for this is largely due to the prevalent demarcation between responsibilities of framework and application developers. For instance an application may use a particular web framework and be unable to negotiate compliance of that framework with OUIA. However they may choose to make the elements that they *do* have control over fully compliant with the OUIA.

2.3 Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in RFC 2119.

2.4 Namespacing

- All HTML elements created for the sole purpose of fulfilling or augmenting this specification **MUST** be prefixed with `ouia`. Currently there are no custom HTML elements defined.
- All custom HTML attributes to be applied to existing HTML elements **MUST** be prefixed with `data-ouia`.
- All additional attributes to existing Javascript objects **MUST** be prefixed with `ouia`.

2.5 Specification Parts

The OUIA specification is split up into several sections. As stated, an application or framework can implement any number of the specification parts, but each section must be implemented completely. Thus a project can be said to be `OUIA:Component` or `OUIA:Page` compliant, as opposed to being completely `OUIA` compliant, but **ALL** components of the `OUIA:Component` part **MUST** be adhered to.

The following blocks exist in the specification:

- `OUIA:Component` - components and component frameworks
- `OUIA:Page` - page identification
- `OUIA:PageSafe` - interaction safety

2.5.1 `OUIA:Component`

This part is designed to cover components in use on pages. Components that are `OUIA:Component` compliant **MUST** have the following properties.

- A root level HTML element with a `data-ouia-component-type` attribute describing a unique name identifying **ALL** HTML components that can be controlled with the same code or interactions. These identifiers **MAY** and **SHOULD** be namespaced when used within a framework, particularly when the name is generic. The delimiter between namespace and type name should be a single `/` character.
  - e.g. A page that has a special dropdown could choose to name that dropdown as `FrameworkA/CustomDropdown`. All instances of this `FrameworkA/CustomDropdown` component **MUST** be expected to be able to be controlled via the same automation.
- An id attribute called `data-ouia-component-id`
  - if there is only one instance of the component on the page at once, it is **OPTIONAL**
  - if there are multiple instances of a component on the page it **MUST** be used
  - it **MUST** be unique within the surrounding context of the component (context may be a surrounding component or collection)
  - e.g. A vertical navigation can be expected to only be instantiated once on a page. As such it does not need any other identifying factors. Another component type, like a button, would be created multiple times on a page and requires some kind of unique identifying id.
  - e.g. An list of items could have tags associated with them. These could be presented in a tabular format. Whilst the ids of those tags could be identical, they are also contained in rows that make them contextually different. An alternative to this is that the id could be prefixed with the item id that it is related to, to create a compound id.
- An attribute called `data-ouia-safe`, which is `True` only when the component is in a static state, i.e. no animations are occurring. At all other times, this value **MUST** be `False`.

2.5.2 `OUIA:Page`

This part is designed to cover page identification. Pages that are `OUIA:Page` compliant **MUST** have the following properties:

- A `<body>` html tag with the following attributes:
  - A `data-ouia-page-type` attribute, which determines the base context of the page.

    * e.g. A page listing some inventory of food items could have `food` as its `data-ouia-page-type` attribute.

  - A `data-ouia-page-action` attribute, which determines if the page has a controller associated with it to perform a specific action. This is **OPTIONAL** if the page does not describe a specific action, or is a single action screen displaying information only.

    * e.g. A page providing the edit action of an inventory item could have an `edit` value for the `data-ouia-page-action` attribute. Whereas a modal about page would not need an `data-ouia-page-action` attribute.

  - A page which dynamically changes action without a page reload **MUST** correctly update the `data-ouia-page-type` and `data-ouia-page-action` attributes if the context of the page changes.

  - An **OPTIONAL** `data-ouia-page-object-id` attribute, where the page is in the context of a specific instance of an object. E.g. A page describing a food item that is being edited, could have an `data-ouia-page-object-id` field of 142526.

- Each of the four main content areas present on the page, *Header*, *Main*, *Navigation*, *Footer*, **MUST** be embellished with the corresponding data attribute to aid in finding components within certain areas.

  - `data-ouia-header="true"` - For the header

  - `data-ouia-footer="true"` - For the footer

  - `data-ouia-main="true"` - For the main content pane

  - `data-ouia-navigation="true"` - For the main navigation pane

  - *Header*, *Footer* and *Main* **MUST NOT** ever be nested inside each other.

  - *Navigation* **MAY** live inside any of the other content areas but must appear only once.

- All links appearing inside the *Navigation* content pane **MUST** specify the `href` attribute with a valid URL to the target, and in addition **MUST** also include a `data-ouia-navigation-name` which identifies the name of the navigation link to aid in building a site map.

- All other links which are considered top level navigational, i.e. they force a page reload/load, **MUST** also be defined with hidden links inside the *Navigation* pane to aid in site map creation.

  - e.g. `<a href="/settings" data-ouia-navigation-name="Settings" hidden="true"/>`

- If OUIA attributes are not enabled by default there **SHOULD** be an HTML local storage variable `ouia:enabled`, to enable the usage of these attributes. Such action **MAY** incur a page restart.

Example of `OUIA:Page`

A page describing the edit action of a food item with the id 142526 could have an attribute looking like `<body data-ouia-page-type="food" data-ouia-page-action="edit" data-ouia-page-object-id="142526">`

2.5.3 `OUIA:PageSafe`

- An attribute named `data-ouia-page-safe` which declares whether the page is safe to access. This should represent the state of all `data-ouia-safe` attributes as well as:

  - Page animations

  - XHR requests

  - Any other operations which affect the DOM structure
- This should take the form of a Boolean, but the specification does not impose a specific implementation.

2.5.4 Example

An example page that fully complies with the OUIA is presented here

2.5.5 Declaration of Conformity

A project wishing to declare its conformity **SHOULD** use one of the supplied badges, or in the case that the entire spec has been conformed to in its entirety should proudly display the overall compliance badge.

2.5.6 License